

National College of Ireland
Higher Diploma in Science in Computing
Software Development
2016/2016

JACEK BYZDRA
X15030491
X15030491@student.ncirl.ie

INTERACTIVE SOFTWARE APPLICATION
FOR DATA STRUCTURE VISUALIZATION
AND ALGORITHM ANIMATION

Technical Report



Declaration Cover Sheet for Project Submission

SECTION 1 Student to complete

Name:

Jacek Byzdra

Student ID:

X15030491

Supervisor:

Dr. Anu Sahni

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Jack Byzdra Date: 04 December 2016

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Table of Contents

Executive Summary	8
1 Introduction	10
1.1 Background.....	10
1.2 Aims.....	11
1.3 Technologies	12
2 System.....	13
2.1 Requirements	13
2.1.1 Functional requirements.....	13
1.1.1 Use Case Diagrams	28
1.1.2 Requirement 73:"Welcome invitation - FR73"	29
1.1.3 Requirement 1:"FR1".....	32
1.1.4 Requirement 2:"FR2".....	35
1.1.5 Requirement 3:"FR3".....	39
1.1.6 Requirement 4:"FR4".....	42
1.1.7 Requirement 5:"FR5".....	45
1.1.8 Requirement 7:"FR7".....	48
1.1.9 Requirement 8:"FR8".....	51
1.1.10 Requirement 9:"FR9".....	52
1.1.11 Requirement 10:"FR10".....	53
1.1.12 Requirement 11:"FR11".....	55
1.1.13 Requirement 12:"FR12".....	57
1.1.14 Requirement 13:"FR13".....	59
1.1.15 Requirement 14:"FR14".....	61
1.1.16 Requirement 15:"FR15".....	62
1.1.17 Requirement 16:"FR16".....	63
1.1.18 Requirement 17:"FR17".....	65
1.1.19 Requirement 18:"FR18".....	67
1.1.20 Requirement 19:"FR19".....	69
1.1.21 Requirement 20:"FR20".....	71
1.1.22 Requirement 21:"FR21".....	73
1.1.23 Requirement 22:"FR22".....	75

1.1.24 Requirement 23:"FR23"	77
1.1.25 Requirement 40:"FR40"	79
1.1.26 Requirement 44:"FR44"	81
1.1.27 Requirement 46:"FR46"	83
1.1.28 Requirement 48:"FR48"	86
1.1.29 Requirement 55:"FR55"	88
1.1.30 Requirement 60:"FR60"	90
1.1.31 Requirement 61:"FR61"	93
1.1.32 Requirement 62:"FR62"	95
1.1.33 Requirement 63:"FR63"	98
1.1.34 Requirement 64:"FR64"	100
1.1.35 Requirement 79:"FR79"	103
2.1.2 Data requirements.....	105
2.1.3 User requirements.....	110
2.1.4 Environmental requirements	110
2.1.5 Usability requirements.....	111
2.2 Design and Architecture.....	112
2.2.1 Design and Architecture – Welcome Section	114
2.2.2 Design and Architecture – Login Section	121
2.2.3 Design and Architecture – Register Section.....	126
2.2.4 Design and Architecture – Menu Section	129
2.2.5 Design and Architecture – Bars Sorts Section	137
2.2.6 Design and Architecture – Panels Sorts Section.....	141
2.2.7 Design and Architecture – Tree Section.....	146
2.2.8 Design and Architecture – Graph Section	151
2.3 Implementation	156
2.3.1 Implementation – main algorithm presentation – graph.....	176
2.3.2 Implementation – main algorithm presentation – sorting algorithm with animation.....	177
2.3.3 Implementation – main algorithm presentation – Binary Search Tree presentation	178
2.3.4 Implementation – class DBConnection.java	179
2.3.5 Implementation – class WelcomeApp.java.....	180
2.3.6 Implementation – class LoginController.java.....	183
2.3.7 Implementation – class LoginUser.java.....	183

2.3.8	Implementation – class User.java.....	185
2.3.9	Implementation – class RegisterController.java	185
2.3.10	Implementation – class WelcomeController.java.....	186
2.3.11	Implementation – class MenuController.java.....	188
2.3.12	Implementation – class HomeController.java	190
2.3.13	Implementation – class SortController.java	193
2.3.14	Implementation – class Sort.java.....	196
2.3.15	Implementation – class TreeBSTViewController.java.....	199
2.3.16	Implementation – class BSTNode.java.....	201
2.3.17	Implementation – class BST.java	201
2.3.18	Implementation – class GraphController.java.....	201
2.3.19	Implementation – class Graph.java	203
2.3.20	Implementation – class GraphEdge.java.....	204
2.3.21	Implementation – class GraphNode.java.....	204
2.3.22	Implementation – class DescriptionController.java.....	204
2.3.23	Implementation – class QuizController.java	204
2.3.24	Implementation – class QuizQuestion.java	205
2.3.25	Implementation – class AlgorithmsDescription.java	205
2.4	Testing	206
2.5	Graphical User Interface (GUI) Layout.....	214
2.6	Customer testing.....	214
2.7	Evaluation	214
2.8	SWAT	215
3	Conclusions	217
4	References	218
5	Appendix.....	219
5.1	Project Proposal	219
5.2	Project Plan	225
5.3	Requirements Specification	225
5.4	Project Analysis & Design.....	322
5.5	Project Test Plans.....	331
5.6	Monthly Journals.....	340

5.6.1	Monthly Journal #1	340
5.6.2	Monthly Journal #2	348
5.6.3	Monthly Journal #3	Błąd! Nie zdefiniowano zakładki.
5.7	Other Material Used.....	Błąd! Nie zdefiniowano zakładki.

Executive Summary

The project presents an interactive software application designed to visualization of commonly used data structures and algorithms. This environment can be used as very useful educational tool in Computer Science. The system has the ability to display data structures and algorithms provide graphical visualization and animation of the tested structure. The first part and the second presents visualizations and animations of different sorting methods in a dynamic way. In the first part of the application, the user can enter the 1000 numbers graphically depicting the bars and can assess the effectiveness of the algorithms. In the second part of the application the user can enter up to 46 numbers represented by graphic panels. This part has been enriched with the slider speed sorting. The user can speed up the sorting algorithm animation and watch the step-by-step, how the algorithms are executed. The application was developed in order to present graphically step by step, how various techniques algorithms are executed. The tool compares a visual way sorting method and measures the execution time of each algorithm. User can enter different size numbers and see how different methods of sorting are done, and compare the execution time of these methods.

The third part of the program shows the visualization of the structure of a binary search tree. The application creates and draws the tree according to the number entered by the user and provides a graphical structure of the binary tree, with nodes. The user can select different ways to search for a binary tree, then the system displays the found solution in the form of text and graphics.

The fourth part of the program shows the visualization of the structure of the graph. The application creates according to the number entered by the user, graphical structure of the graph, with nodes and edges. The user can select the start and end node in the graph created and the search for the shortest ways paths according to Dijkstra's algorithm. When a solution is found, the system displays a graphical and textual solution, and shows the shortest paths and ways, as well as the nodes and paths. An additional extension of the system is the

presentation of text information of different algorithms. The system displays the notes when the selected button is triggered by the user. Welcome page of the program shows an animated logo of the system. User can register to the system if the login has not been created before. User database was created in a MySQL database. Login is protected and login is required to access the system.

The fifth element of the system was designed to provide the user with a variety of quiz questions and answers, the content of which is based on algorithms and data structures. You can start a quiz and choose the answer from the list. The system will evaluate user answers and will show the correct answers and the total points received.

The list of questions and answers are stored in a MySQL database.

The evaluation was carried out by a student studying in the National Colledge of Ireland.

1 Introduction

This document was prepared as a main project for the Higher Diploma in Science in Computing in Software Development specialization.

The theme of the project is 'INTERACTIVE SOFTWARE APPLICATION FOR DATA STRUCTURE VISUALIZATION AND ALGORITHM ANIMATION'. The present document provides the basis for the origin of the program and describes its evolution in recent months.

1.1 Background

The project presents an interactive software application designed to visualization of commonly used data structures. One of the application feature is animation of used algorithms.

Animation is an innovative way that provides an exciting and dynamic platform to encourage students to give interesting and engaging presentation.

This is a new tool for learning difficult technical skills. One of the goals is to provide a tool animation cognitive function, which consequently allows students understanding of the subject. Animation is used to demonstrate concepts and things visually, showing how things work together.

This attracts the attention of students, encouraging active learning and sustains motivation. Emphasizes the development of students' skills and rational thinking, enables and encourages students to explore and create solutions to the problem.³

Visualization is a technique for creating images, diagrams, or animations to communicate a message. Visualization of the text is a way to improve reading comprehension. It refers to our ability to imagine the images based on text you are reading. Students who visualize how reading can remember what they have read for longer periods of time. Visualization through visual imagery is an effective way to communicate abstract and concrete ideas. Visualization algorithms graphically shows how the algorithms work.

This can help students understand the basic operations of algorithms in computer science.^{4,5}

The project presents both animations and visualizations of typical algorithms and data structures in computer science.

The ability to use animated learning systems caused a lot of efforts from the IT community to the development of web-based systems animated.

Compared to other solutions, the project was developed in an innovative way. Visualization and animation presented here is a set of algorithms and data structures with interactive user interface. In addition, the application contains a quiz tool in the field of data structures and algorithms.

1.2 Aims

The project presents an interactive software application designed to visualization of commonly used data structures. One of the application feature is animation of used algorithms. The tool is dedicated to the study of the main data structures by use interactive interface to guide in Java programming language. The application can be used in training courses presenting commonly used data structures and algorithms. This environment can be used as very useful educational tool in Computer Science.

The algorithms and data structures are key in majority of computer programs.

The tool was developed to help :

- Presentation of the concepts of algorithms and data structures in an animated and graphical form.
- Presentation of an interactive feature that allows students to change the methods and data, and compare the effectiveness of different algorithms in a visual way.
- Facilitate the student's grasp programming of common data structures and algorithms,

- Presentation of a feature that allows to change data and methods, and observe textual and visual solution in different data structures.
- Presentation of an interactive check quiz tool for self evaluation in the field of algorithms and data structures.

1.3 Technologies

A variety of technologies were utilised during the development of the app.

The short list of technologies used in the program is described below::

- Java JDK, JRE 1.8.0_101 was used as the primary language. The project was developed parallelly in an Eclipse Neon 4.6.1 and Netbeans IDE 8.1,
- JavaFX SDK 8: JavaFX compiler and runtime tools and 2D graphics and media libraries to create highly interactive applications for desktop application.
- Java FX FXML: JavaFX Script, designed to write code that closely corresponds to the system GUI.
- JavaFX Scene Builder 2.0: Layout visual environment for fast, easy and convenient user interface design.
- MySQL 5.7 database
- MySQL Workbench 6.3.6 management of the database
- MySQL JDBC driver for MySQL: mysql-connector-java-5.1.39-bin.jar

2 System

This section provides details information about requirements of the system.

2.1 Requirements

The project requirements are conditions , and tasks, that had to be completed to ensure the completion and success of the project.

This section provides picture of the work that had to be done to achieve completion of the project.

2.1.1 Functional requirements

The functional requirements specifies the operations and activities that a system must be able to perform.

These system requirements are categorised by its necessity for the app to operate.

- Cat1: Mandatory – The app will not serve its primary function in the absence of this requirement;
- Cat2: Ideal – Adds value to the app but does not impact on its primary function or operation;
- Cat3: Novel – Increases the attraction of the app but does not provide any functionality.

The list of the required functional requirements of the system is presented below:

No	Description of Functional requirement	When it is required ?	Dependencies	Category
FR1	Visual Animation of Bubble Sort Algorithm (Sorts Bars View). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Bubble Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1

FR2	Visual Animation of Insertion Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR3	Visual Animation of Selection Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR4	Visual Animation of Merge Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Merge Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR5	Visual Animation of Quick Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Quick Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR6	Show time of sorting operation in Sorts Bars Window	When the program runs in Sorts Bars Window, and user press the sort button for each algorithm. System displays time of the opearation when sortinmg is finished	System is running, User is logged in, Sorts Bars window is opened, User invoked sorting of any algorithm before	1
FR7	Switch to Menu window from Sorts Bars window after home button is pressed	When the program runs and Sorts Bars window is opened. System should switch to Menu window independently of running sorting algorithms	User is logged in, Sorts Bars window is opened, System is running and Sorts Bars Window is opend	1

FR8	Maximize Window when maximize button is pressed	When the program runs , and is not in a register mode	System is running. Register window is not opened	2
FR9	Minimize Window when minimize button is pressed	When the program runs , and is not in a register mode	System is running. User is logged in, Register window is not opened	2
FR10	Generate Data for sorting bars when the generate button is pressed	When the program runs and Sorts Bars window is opend, user provided correct integer number to sort before, and user pressed generate data button	System is running, User is logged in, and Sorts Bars Window is opend, user provided correct integer number to sort label	1
FR11	Display information "Incorrect sort number"	When the program runs and Sorts Bars window is opend, user provided incorrect integer number to sort before, and user pressed generate data button	System is running, user is logged in and Sorts Bars Window is opend, user provided incorrect integer number to sort label	1
FR12	Display Buttons Generate Data, Home, and Integer Sort Label in Sorts Bars window	When the program runs in Sorts Bars Window	System is running., user is logged in and the Sorts Bars window is opened	1
FR13	Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window	When the program runs in Sorts Bars window, and user entered and generated correct integer number to sort	When the program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort	1
FR14	Display information about logged user in Sorts Bars window	When the program runs , user is logged in, and Sorts Bars window is opend	When the program runs , user is logged in, and Sorts Bars window is opend	1
FR15	Log user out after Logout button is pressed	When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button	When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button	1

	Visual Animation of Bubble Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Bubble Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	
FR16	Visual Animation of Insertion Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR17	Visual Animation of Selection Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Selection Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR18	Visual Animation of Heap Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Merge Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR19	Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Quick Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR20	Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Quick Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR21	Switch to Menu window from Sorts Panel window after home button is pressed	When the program runs and Sorts Panels window is opened. System should switch to Menu window independently of running sorting algorithms	User is logged in, Sorts Panels window is opened, System is running and Sorts Panels Window is open	1

	Generate Integer Data for sorting bars when the generate button is pressed	When the program runs and Sorts Panels window is open, user provided correct integer number to sort before, and user pressed generate data button	System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label	
FR22	Display information "Incorrect sort number"	When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button	System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label	1
FR23	Display Buttons Generate Data, Home, and Integer Sort Label in Sorts Panels window	When the program runs in Sorts Panels Window	System is running., user is logged in and the Sorts Panels window is opened	1
FR24	Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Panels window	When the program runs in Sorts Panels window, and user entered and generated correct integer number to sort	When the program runs, user is logged in in Sorts Panels window, and user entered and generated correct integer number to sort	1
FR25	Display information about logged user in Sorts Panels window	When the program runs , user is logged in, and Sorts Panels window is open	When the program runs , user is logged in, and Sorts Panels window is open	1
FR26	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Sort Bars window is open, and user pressed Logout button	When the program runs , user is logged in, and Sorts Bars window is open, and user pressed Logout button	1
FR27	Visual Creation of Binary Search Tree from generated integer data	When the program runs in Binary Search Tree window, user provided before corrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided correct number to BST search label	1
FR28				

	Visual Animation of Inorder Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Inorder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	
FR29	Visual Animation of Post Order Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed PostOrder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR30	Visual Animation of Pre Order Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed PreOrder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR31	Visual Animation of Search Leaves Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Leaves button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR32	Visual Animation of Search Max value of nodes Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Max button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR33	Visual Animation of Search Max value of node in BST(Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Max button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR34	Visual Animation of Search of Min value of node in BST (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Min button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	
FR35				

	Display textual information of Height of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Height button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	
FR36	Textual display of size of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Size button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR37	Textual display of depth of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Depth button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR38	Generate number of nodes in the Binary Search Tree when the generate data button is pressed	When the program runs in Binary Search Tree window, user provided before corrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided correct number to BST search label	1
FR39	Display information "Enter correct number of nodes"	When the program runs in Binary Search Tree window, user provided before incorrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided incorrect number to BST search label	1
FR40	Display buttons Generate Data, Home, Inorder, PostOrder, PreOrder, leaves, Max, Min, Height, Size, Depth, Label number of nodes in Binary Search Tree window	When the program runs in Binary Search Tree window,	System is running, user is logged in, Binary Search Tree window is opened	1
FR41				

FR42	Display information about logged user in Binary Search Window	When the program runs , user is logged in, and Binary Search Tree window is open	System is running, user is logged in, Binary Search Tree window is open	1
FR43	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Binary Search Tree window is open, and user pressed Logout button	When the program runs , user is logged in, and Binary Search window is open, and user pressed Logout button	1
FR44	Switch to Menu window from Binary search Tree window after home button is pressed	When the program runs and Binary Search Tree window is opened. System should switch to Menu window after home button is pressed independently of running search methods in BST window	User is logged in, Binary Search Tree window is open, System is running	1
FR45	Visual Creation of Graph from generated integer data	When the program runs in Graph window, user provided before correct number of nodes to generate, and pressed Generate Graph button	System is running, user is logged in, Graph window is open , user provided correct number to graph search label	1
FR46	Switch to Menu window from Graph window after home button is pressed	When the program runs and Graph Panels window is opened. System should switch to Menu window independently of running graph searching algorithms	User is logged in, Graph window is open, System is running	1
FR47	Generate Integer Data for creating graph when the generate button is pressed	When the program runs and Graph Panels window is open, user provided correct integer number to generate	System is running, User is logged in, and Graph Window is open, user provided correct integer number to graph label	1

		graph before, and user pressed generate graph button		
FR48	Display information "Incorrect graph size If you want generate graph please enter correct graph size"	When the program runs and Graph window is open, user provided incorrect integer number to create graph before, and user pressed generate graph button	System is running, user is logged , Graph Window is open, user provided incorrect integer number to graph label	1
FR49	Display Buttons Generate Graph Home, and Integer Graph Label in Graph window	When the program runs in Graph Window	System is running., user is logged in and the Graph window is opened	1
FR50	Display Button:Start Node Label, End Node Label, Set start/end button, Dijkstra's Shortest Paths button in Graph window	When the program runs inGraph window, and user entered and generated correct integer number to create graph	When the program runs, user is logged in in Graph window, and user entered and generated correct integer number to create graph	1
FR51	Display information about logged user in Graph window	When the program runs , user is logged in, and Graph window is open	When the program runs , user is logged in, and Graph window is open	1
FR52	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Graph window is open, and user pressed Logout button	When the program runs , user is logged in, and Sorts Bars window is open, and user pressed Logout button	
FR53	System visualize the start, end node, and the shortest path in different color. Displays textual information about number of nodes in Graph, number of edges in Graph, Displays textual information about sum of weights of paths from start Node to every Node. Next system displays textual information abot the shortest path from start to end node,	When Dijkstra's Shortest Paths button is pressed System visualize the start, end node, and the shortest path in different color, displays textual information about number of nodes in Graph, number of edges in Graph, Displays textual information about sum of	When the program runs, user is logged in, and Graph window is opened, and user provided correct number of nodes to create graph, and press button generate graph	1

	information about weights of the shortest path, and information about which Nodes are in the shortest path.	weights of paths from start Node to every Node. Next system displays textual information about the shortest path from start to end node, information about weights of the shortest path, and information about which Nodes are in the shortest path.		
FR54	Visualize start , end node in Graph window	When graph is created and correct start, end node value is provided to label graph and set start/end button is pressed system displays start , end node in different color	When the program runs, user is logged in, and Graph window is opened, and user provided correct number of nodes to create graph, and press button set start/end node in graph	1
FR55	Display information: please enter correct index for start node and for end node and click Set button, when the provided value of start end node was incorrect	When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button	When the program runs, user is logged in, and Graph window is opened, and user provided incorrect number of nodes to create graph, and press button set start/end node in graph	1
FR56	Display Buttons: Quiz, Description, Sorts Bars, Sorts Panels, Trees, Graphs, Hints: Bubble Sort, Hints, logged user in Menu window	When the program runs in Menu Window	System is running., user is logged in and the Menu window is opened	1
FR57	Display Button:Hints: Insertion Sort, Hints Selection Sort, Hints merge Sort, Hints Quick Sort, Hints Trees, Hints Graph in Menu window	When the program runs in Menu Window and Hints button is pressed	When the program runs, user is logged in, and Hints button is pressed	1

FR58	Display information about logged user in Menu window	When the program runs , user is logged in, and Menu window is opend	When the program runs , user is logged in, andMenu window is opend	1
FR59	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Menu window is opend, and user pressed Logout button	When the program runs , user is logged in, and Menu window is opend, and user pressed Logout button	1
FR60	Switch to Sorts Bars window from Menu Window after Sorts Bars button is pressed	When the program runs and Menu window is opened. System should switch to Sorts Bars window after Sorts Bars button is pressed	User is logged in,Menu window is opened, System is running	1
FR61	Switch to Sorts Panels window from Menu Window after Sorts Panels button is pressed	When the program runs and Menu window is opened. System should switch to Sorts Panels window after Sorts Panels button is pressed	User is logged in,Menu window is opened, System is running	1
FR62	Switch to Tree window from Menu Window after Tree button is pressed	When the program runs and Menu window is opened. System should switch to Tree window after Tree button is pressed	User is logged in,Menu window is opened, System is running	1
FR63	Switch to Graph window from Menu Window after Graph button is pressed	When the program runs and Menu window is opened. System should switch to Graph window after Graph button is pressed	User is logged in,Menu window is opened, System is running	1
FR64	Switch to Quiz window from Menu Window after Quiz button is pressed	When the program runs and Menu window is opened. System should switch to Quiz window after Quiz button is pressed	User is logged in,Menu window is opened, System is running	1

	Display information about system when Description button is pressed	When the program runs and Menu window is opened System should display information about the system	User is logged in,Menu window is opened, System is running	1
FR65	Display Hints information about Bubble Sort algorithm when Hints Bubble Sort button is pressed	When the program runs in Menu Window and Hints Bubble Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR66	Display Hints information about Insertion Sort algorithm when Hints Insertion Sort button is pressed	When the program runs in Menu Window and Hints Bubble Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR67	Display Hints information about Selection Sort algorithm when Hints Selection Sort button is pressed	When the program runs in Menu Window and Hints Selection Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR68	Display Hints information about Merge Sort algorithm when Hints Merge Sort button is pressed	When the program runs in Menu Window and Hints Merge Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR69	Display Hints information about Quick Sort algorithm when Hints Quick Sort button is pressed	When the program runs in Menu Window and Hints Quick Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR70	Display Hints information about BST Trees data structure when Hints Trees button is pressed	When the program runs in Menu Window and Hints Trees button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR71	Display Hints information about Graphs data structure when Graphs button is pressed	When the program runs in Menu Window and Hints Graphs button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR72	Display Welcome Window with animation and visualization of system logo, and display Enter button when the application is started	When user starts the application.	System is not started yet.	1
FR73				

	Display information about author of the program in top of each window , except Register window	When system is running the information about author of the program should be displayed in each window except register window	System is running, register Window is not opened	2
FR74	Switch to Login Window from Welcome Window when Enter button is pressed in Welcome Window	When the program runs in Welcome Window and Enter button is pressed	Program runs in Welcome Window	1
FR75	Display Login Window with login button, user Name label, Password label , and Register button in Login Window	When the program runs in Welcome Window and Enter button is pressed	Program runs in Welcome Window, Enter button is pressed	1
FR76	Switch to Register Window when register button is pressed in Login Window	When the program runs in Login Window and Register button is pressed	Program runs in Login Window, Register button is pressed	1
FR77	Display fields: User Name, Password, Confirm Password, E-mail, Confirm E-mail, and button Cancel, and Sign Up in Register Window	When the program runs in Login Window and Register button is pressed	Program runs in Login Window, Register button is pressed	1
FR78	Display information: User name is incorrect. Please enter correct user name for register when the user name was not provided in registration form in Registration Window and Sign Up button was pressed	When the program runs in Register Window and user not provided user name and pressed Sign Up button	Program runs in Register Window	1
FR79	Display information: Password is empty when password was not provided to the system, after user name filled user name field and pressed Sign Up button in Register Window	Display information: Password is empty when password was not provided to the system, after user name filled user name field and pressed Sign Up button in Register Window	Program runs in Register Window	1
FR80				

	Display information: Incorrect Password confirmation when Confirm Password was not provided correctly to registration form, after user name, and password was filled and Sign up button was pressed in Register Window	Display information: Incorrect Password confirmation when Confirm Password was not provided correctly to registration form, after user name, and password was field and Sign up button was pressed in Register Window	Program runs in Register Window	1
FR81	Display information: Email address is empty when Email address was not provided to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Registration Window	Display information: Email address is empty when Email address was not provided to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Registration Window	Program runs in Register Window	1
FR82	Display information: Incorrect Email address when Email address was not provided correctly to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Register Window	Display information: Incorrect Email address when Email address was not provided correctly to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Register Window	Program runs in Register Window	1
FR83	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, after user name, password, confirmation password, and email was filled	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, after user name, password,	Program runs in Register Window	1
FR84				

	and Sign up button was pressed in Register Window	confirmation password, and email was filled and Sign up button was pressed in Register Window		
FR85	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, or provided incorrectly, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, or provided incorrectly, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Program runs in Register Window	1
FR86	Display information: You are registered in this application Please log in now when all user provided correctly all required fields in registration form and pressed Sign Up button in registration window	Display information: You are registered in this application Please log in now when all user provided correctly all required fields in registration form and pressed Sign Up button in registration window	Program runs in Register Window	1
FR87	Switch to Login Window from registration form when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Switch to Login Window from registration form when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Program runs in Register Window	1
FR88	Switch to Login Window from registration form when user pressed cancel button in	Switch to Login Window from registration form when user pressed cancel	Program runs in Register Window	1

	Registration Window	button in Registration Window		
FR89	Register user in MySQL database visualdatajb when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Register user in MySQL database visualdatajb when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Program runs in Register Window, user filled correctly all required fields in registration form in registry Window and pressed Sign Up button	1
FR90	Display information: User/Password combination is not valid. Are you new ? Please use register option, when username or password was not provided or provided not correctly to login fields in LoginWindow	Display information: User/Password combination is not valid. Are you new ? Please use register option, when username or password was not provided or provided not correctly to login fields in LoginWindow	Program runs in Login Window	1

1.1.1 Use Case Diagrams

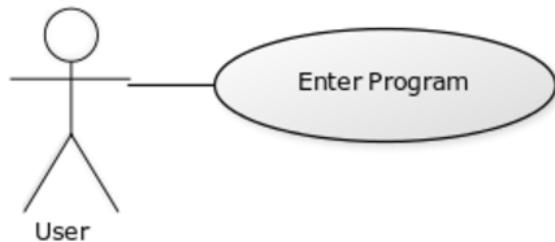


Figure 1 Use Case Diagram FR73

1.1.2 Requirement 73: "Welcome invitation - FR73"

1.1.2.1 Description & Priority

When the user starts to use the system it will display a welcome screen, with the enter button. The welcome screen animates logo and title of the welcome window.

1.1.2.2 Use Case

Scope

The scope of this use case is to invite the user.

Description

This use case describes the process of invitation.

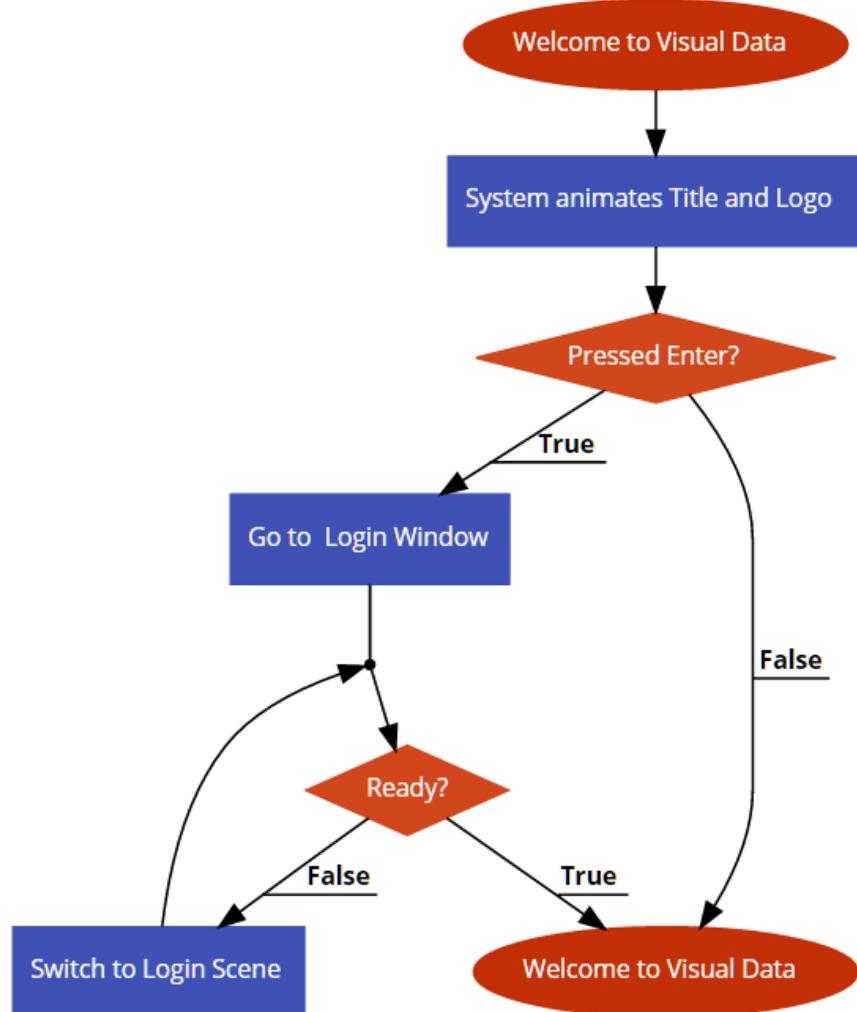


Figure 2 Flow Diagram FR73

Termination Flow Description

Precondition

The user has opened system.

Activation

This use case starts when a user opens system.

Main flow

- A1. The system starts the application.
- A2. The system animates title and logo

Alternate flow

- B1: The system starts the application.

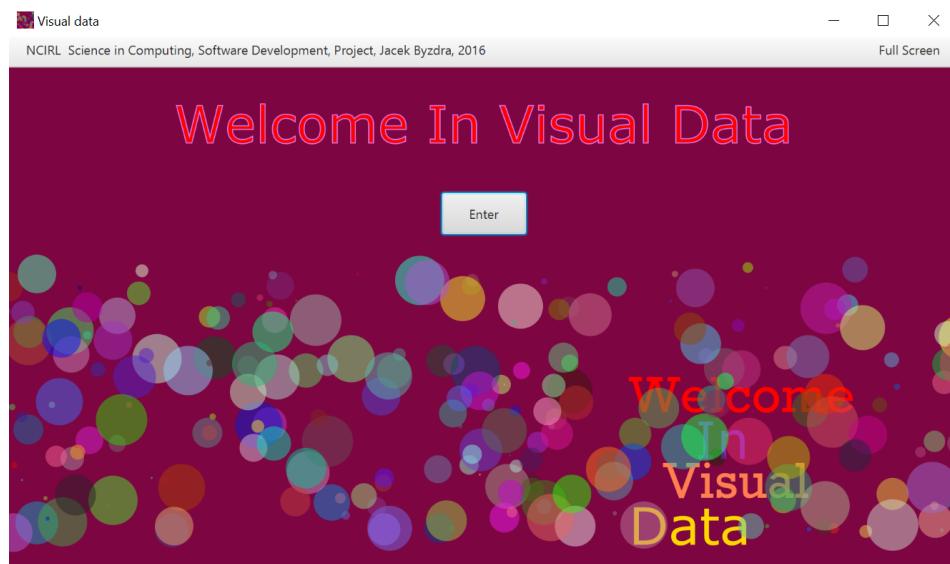
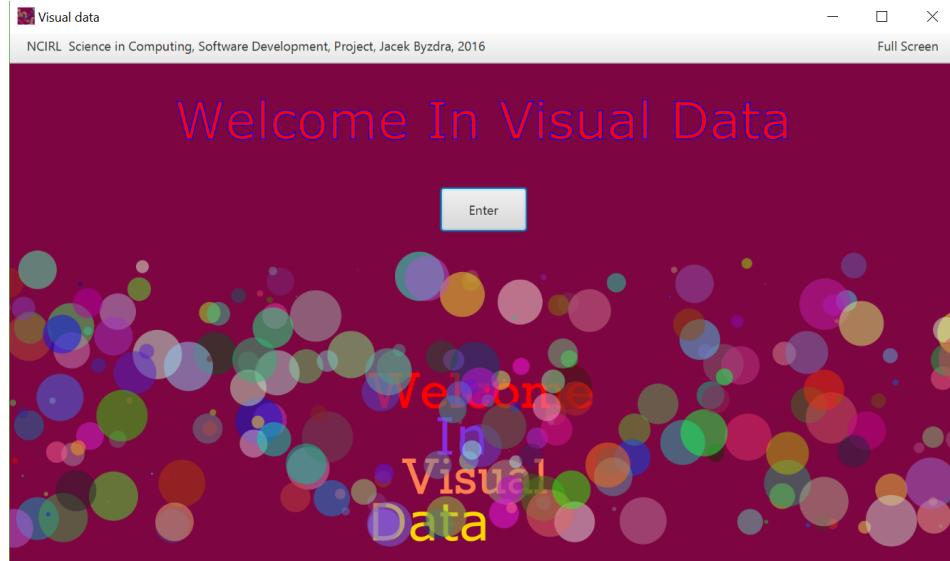
Termination

The user press Enter button

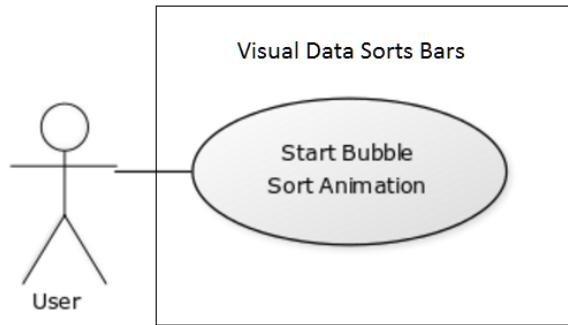
Post condition

The system animates title and logo

Post condition Mock



1.1.3 Requirement 1:"FR1"



1.1.3.1 Description & Priority

Visual Animation of Bubble Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Bubble Sort button. System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

1.1.3.2 Use Case

Scope

The scope of this use case is to start sort Bubble Sort in Sorts Bars view.

Description

This use case describes the process of sorting algorithm Bubble Sort in Sorts Bars window

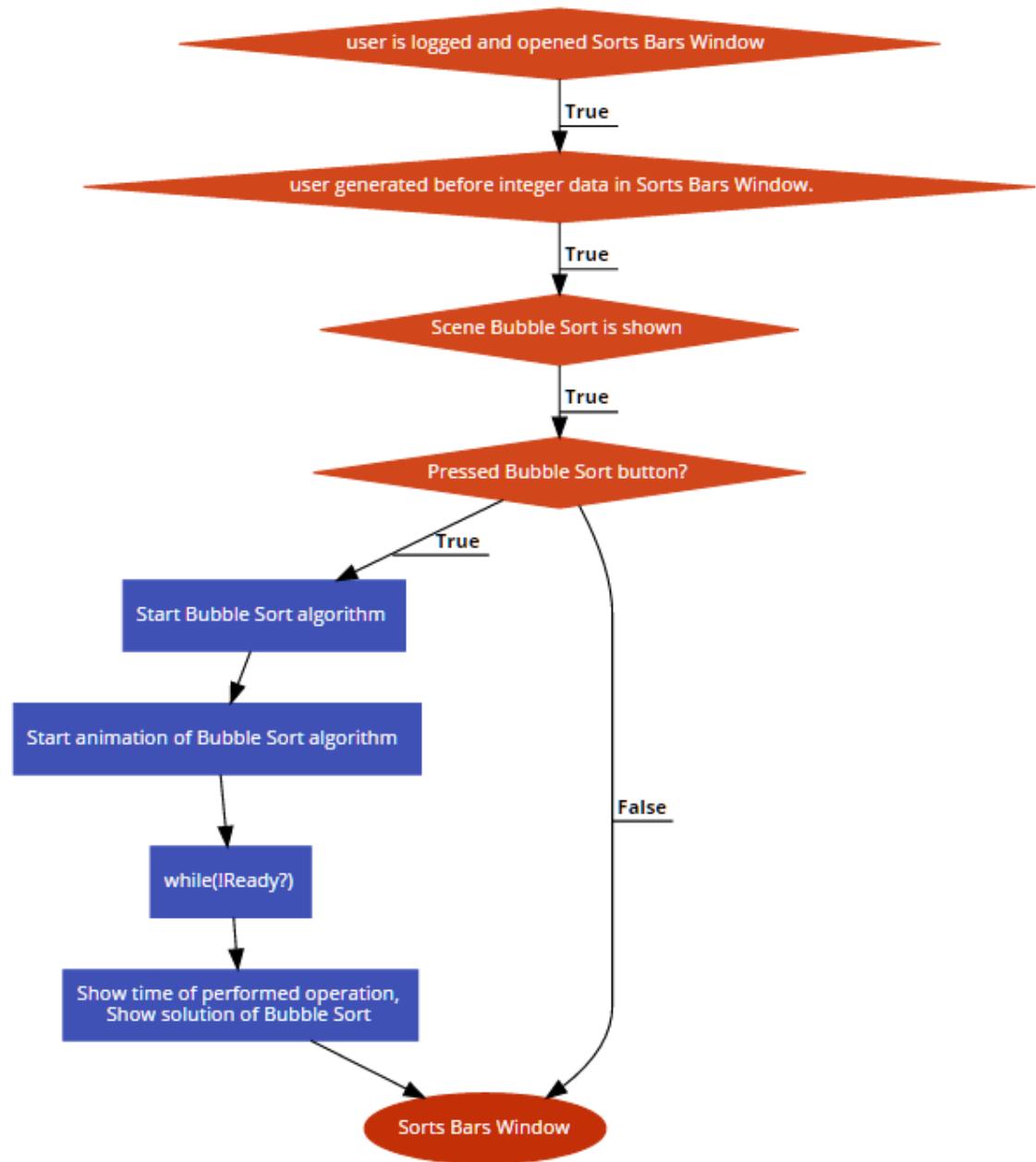


Figure 2 Flow Diagram FR1

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Bubble Sort button.

Main flow

A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Bubble Sort in Sorts Bars window.

A2. The system starts Bubble Sort algorithm.

A3. The system shows animation of Bubble Sort algorithm.

A4. When the Bubble Sort algorithm ends system shows time of performed Bubble Sort operation.

A5. When animation is finished system shows final Bubble Sorts scene in Sorts Bars window with solution.

Alternate flow

B1: The system shows Sorts Bars window.

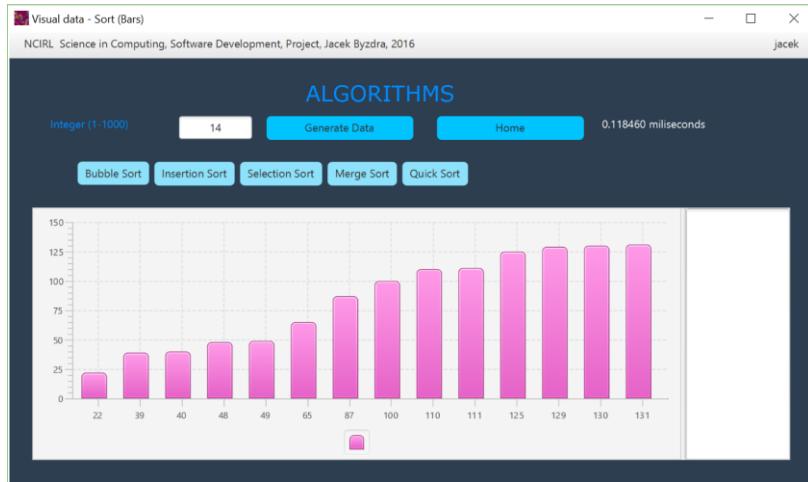
Termination

The user press Enter Bubble Sort button

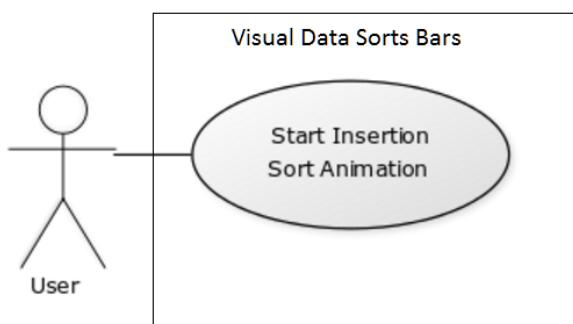
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



1.1.4 Requirement 2: "FR2"



1.1.4.1 Description & Priority

Visual Animation of Insertion Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press

Insertion Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

1.1.4.2 Use Case

Scope

The scope of this use case is to start sort Insertion Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Insertion Sort in Sorts Bars window

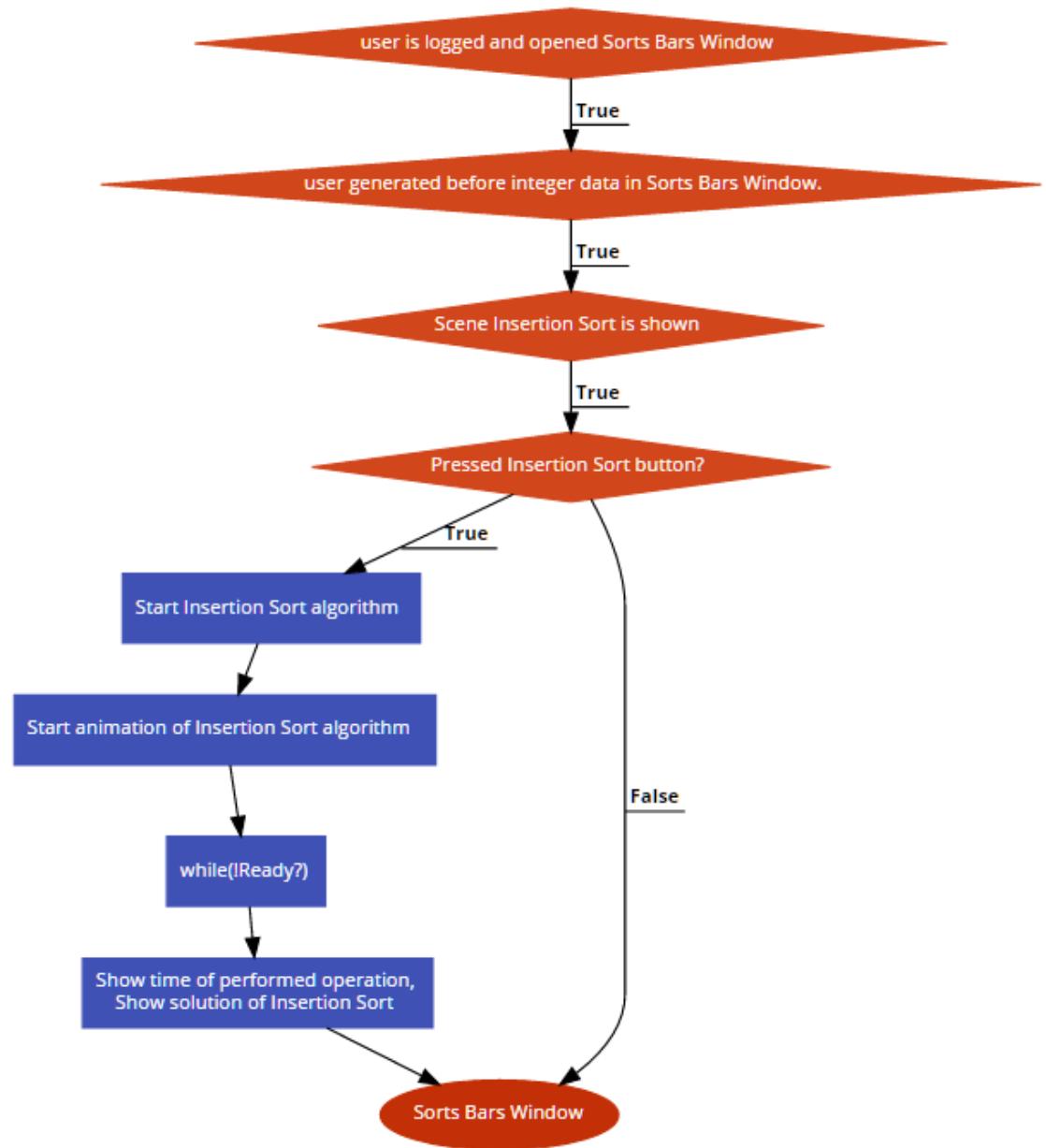


Figure 2 Flow Diagram FR2

Termination Flow Description

Precondition

System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Insertion Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Insertion Sort in Sorts Bars window.
- A2. The system starts Insertion Sort algorithm.
- A3. The system shows animation of Insertion Sort algorithm.
- A4. When the Insertion Sort algorithm ends system shows time of performed Insertion Sort operation.
- A5. When animation is finshed system shows final Insertion Sort scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

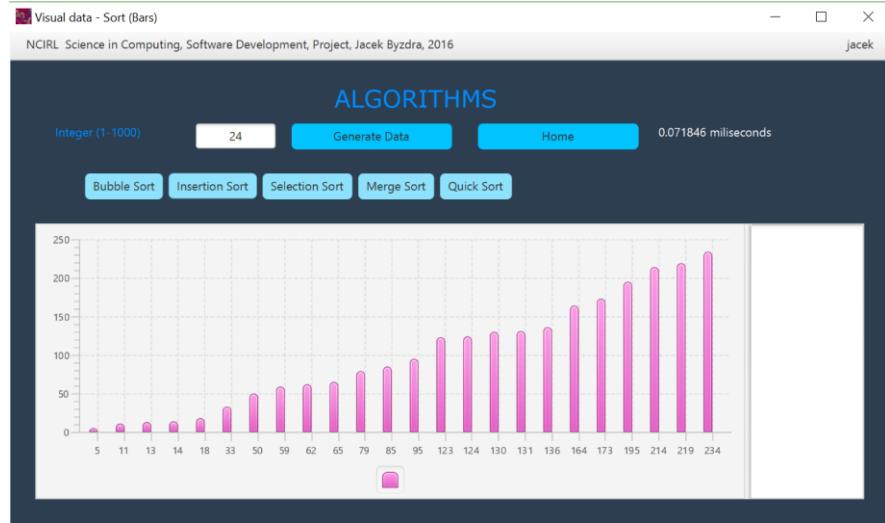
Termination

The user press Enter Isetion Sort button

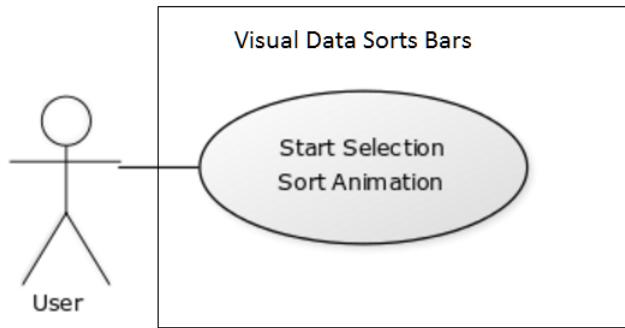
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



1.1.5 Requirement 3:"FR3"



1.1.5.1 Description & Priority

Visual Animation of Selection Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Selection Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

1.1.5.2 Use Case

Scope

The scope of this use case is to start sort Selection Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Selection Sort in Sorts Bars window

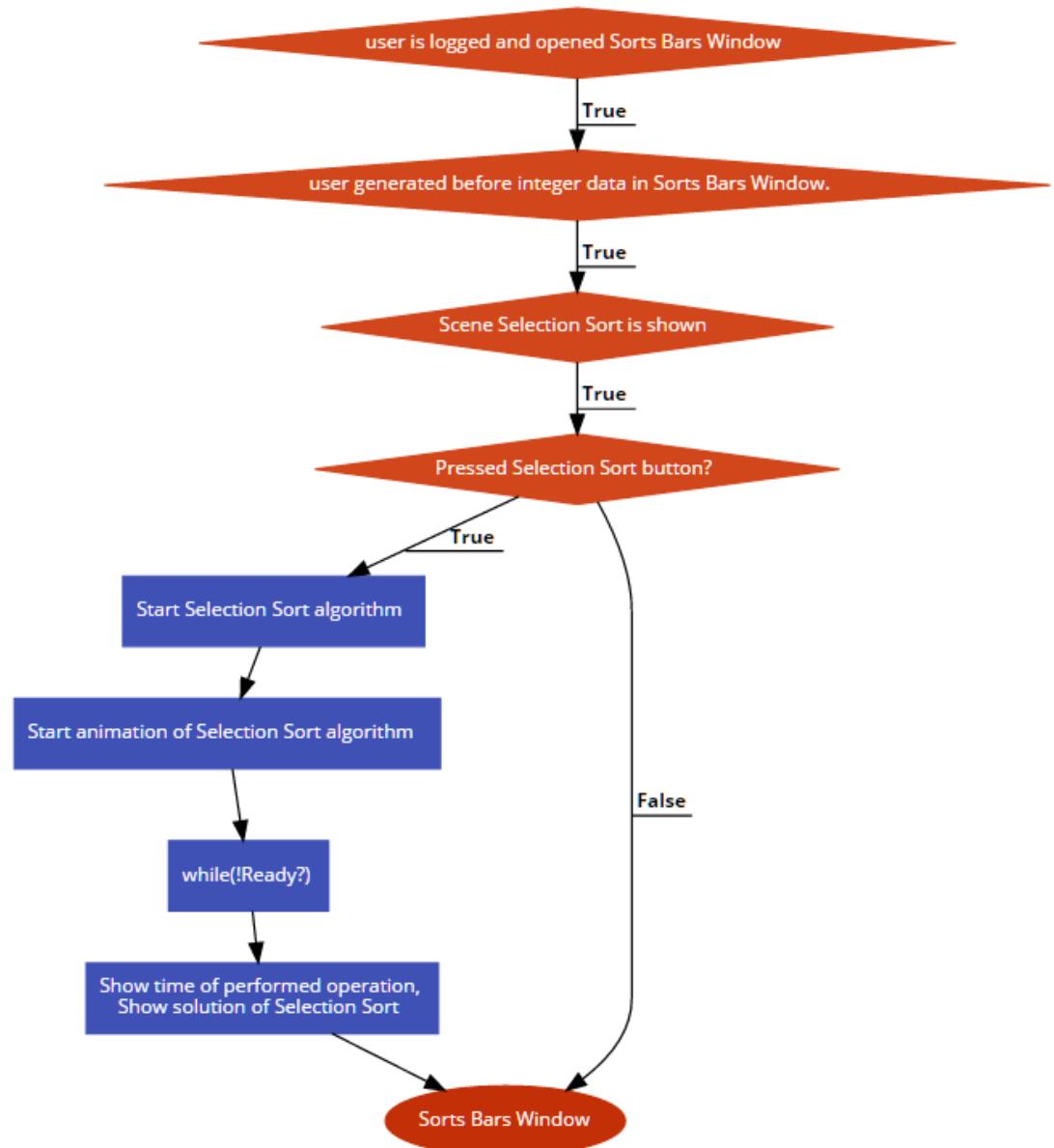


Figure 3 Flow Diagram FR3

Termination Flow Description

Precondition

System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Selection Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Selection Sort in Sorts Bars window.
- A2. The system starts Selection Sort algorithm.
- A3. The system shows animation of Selection Sort algorithm.
- A4. When the Selection Sort algorithm ends system shows time of performed Selection Sort operation.
- A5. When animation is finished system shows final Selection Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

Termination

The user press Enter Selection Sort button

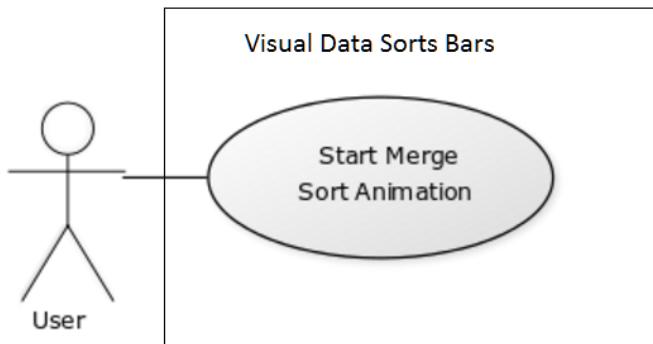
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



1.1.6 Requirement 4: "FR4"



1.1.6.1 Description & Priority

Visual Animation of Merge Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Merge Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

1.1.6.2 Use Case

Scope

The scope of this use case is to start sort Merge Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Merge Sort in Sorts Bars window

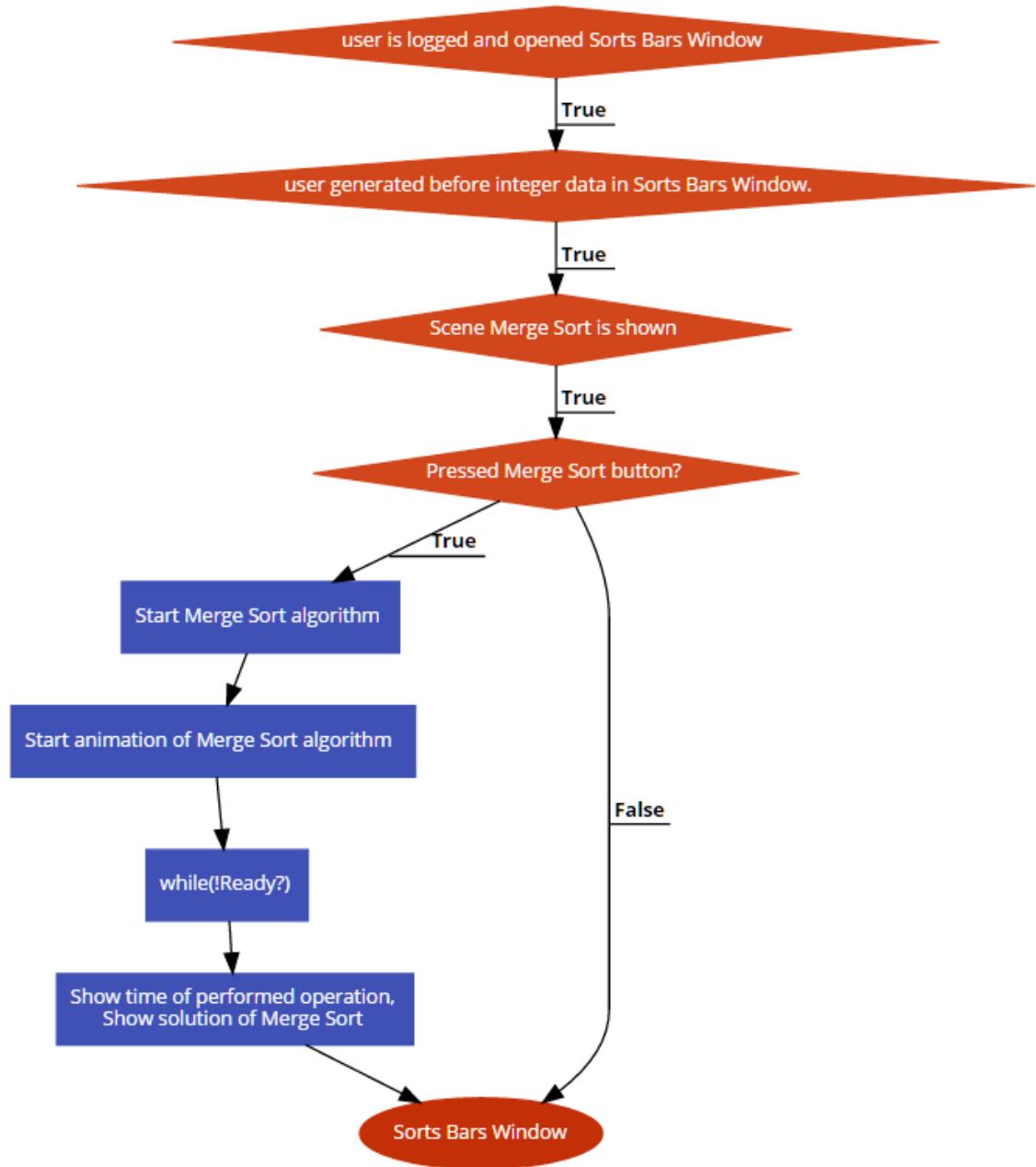


Figure 4 Flow Diagram FR4

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Merge Sort button.

Main flow

A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Merge Sort in Sorts Bars window.

A2. The system starts Merge Sort algorithm.

A3. The system shows animation of Merge Sort algorithm.

A4. When the Merge Sort algorithm ends system shows time of performed Merge Sort operation.

A5. When animation is finished system shows final Merge Sorts scene in Sorts Bars window with solution.

Alternate flow

B1: The system shows Sorts Bars window.

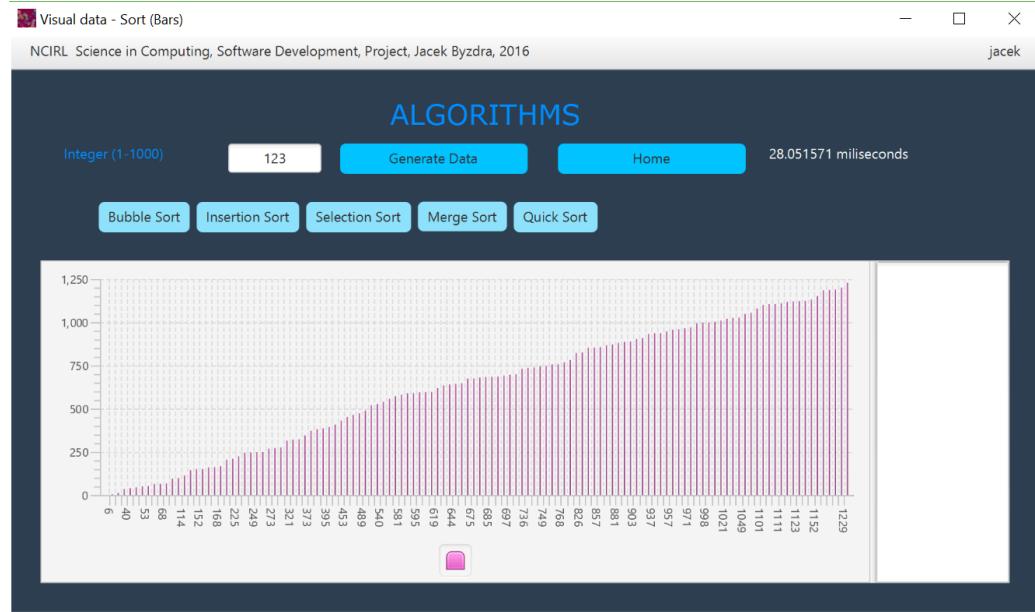
Termination

The user press Enter Merge Sort button

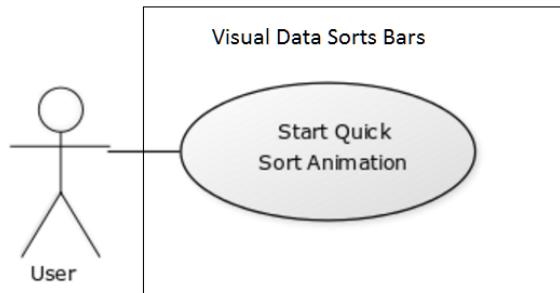
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



1.1.7 Requirement 5:"FR5"



1.1.7.1 Description & Priority

Visual Animation of Quick Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Quick Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

1.1.7.2 Use Case

Scope

The scope of this use case is to start sort Quick Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Quick Sort in Sorts Bars window

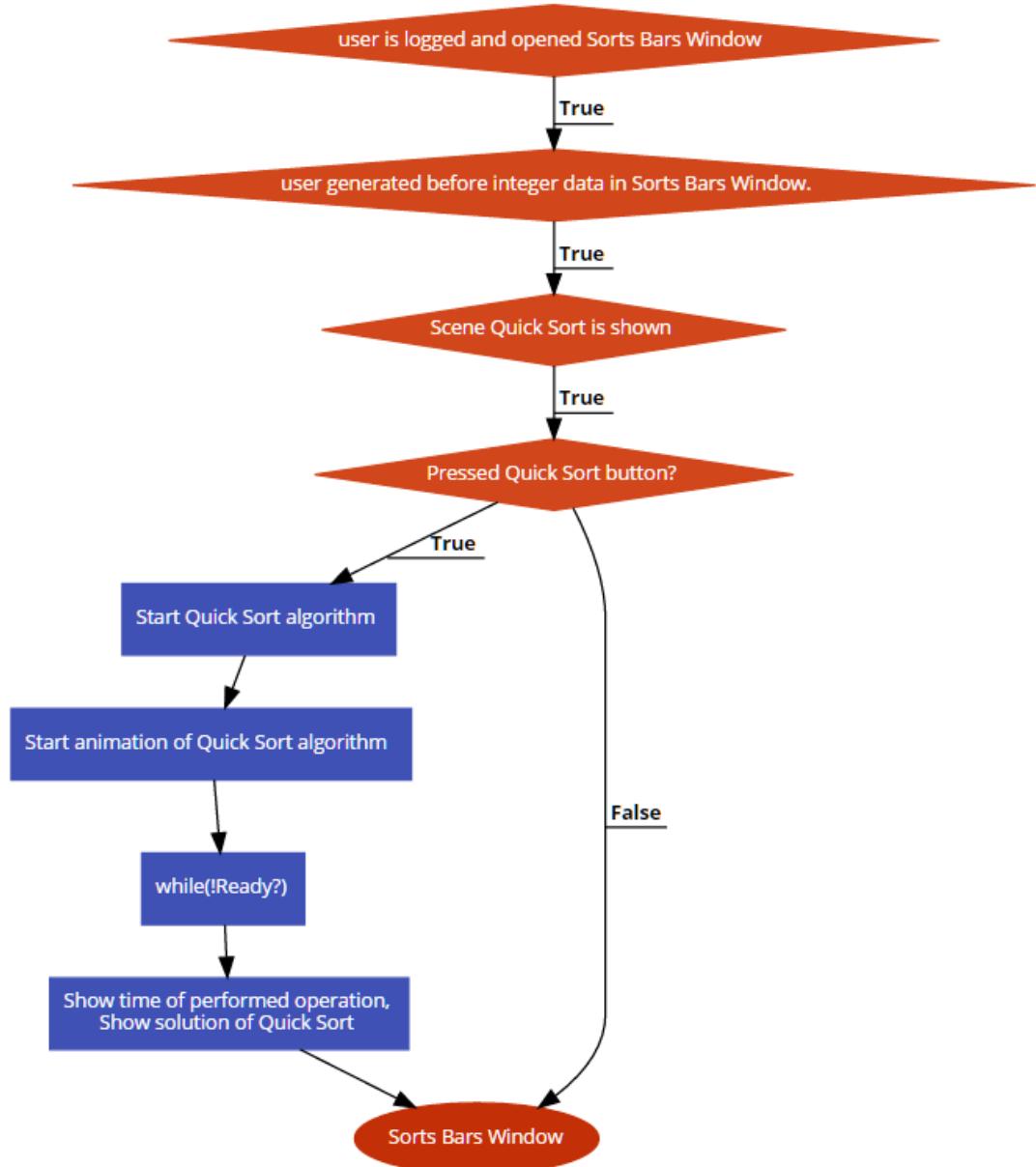


Figure 4 Flow Diagram FR4

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Quick Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Quick Sort in Sorts Bars window.
- A2. The system starts Quick Sort algorithm.
- A3. The system shows animation of Quick Sort algorithm.
- A4. When the Quick Sort algorithm ends system shows time of performed Quick Sort operation.
- A5. When animation is finished system shows final Quick Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

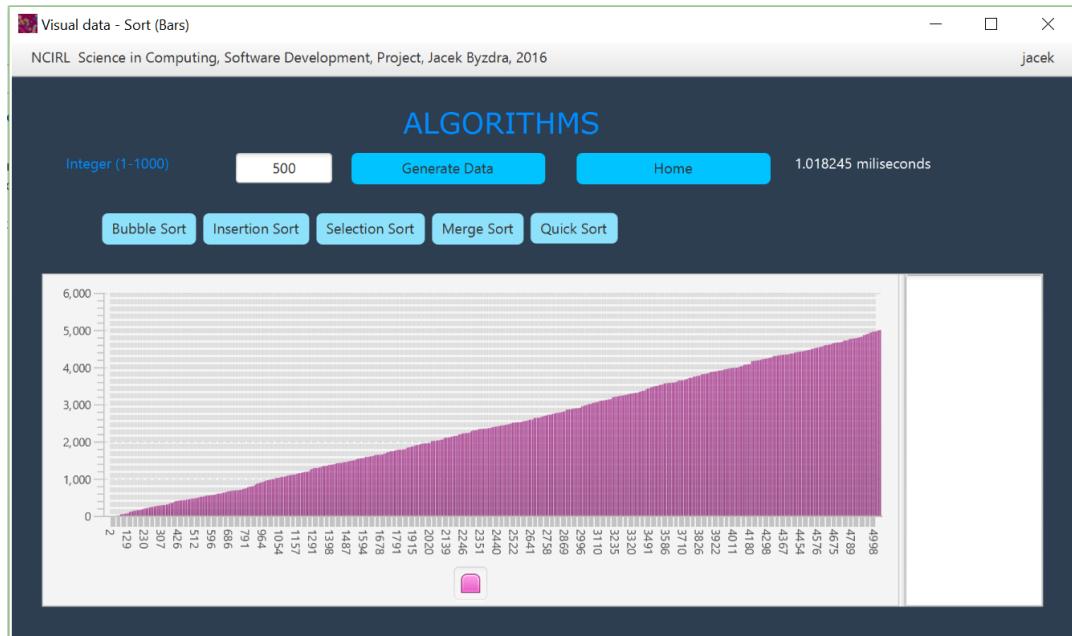
Termination

The user press Enter Quick Sort button

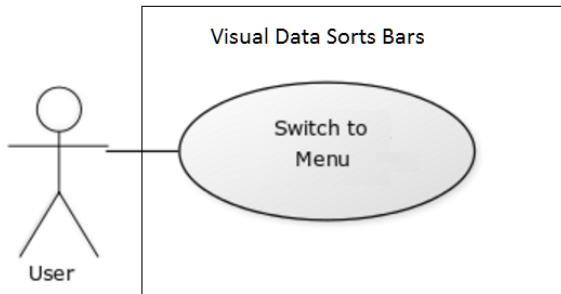
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



1.1.8 Requirement 7:"FR7"



1.1.8.1 Description & Priority

Switch to Menu window from Sort Bars window after home button is pressed. When the program runs and Sorts Bars window is opened. System should switch to main window independently of running sorting algorithms. User is logged in, System is running and Sorts Bars Window is open.

1.1.8.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Sorts Bars window.

Description

This use case describes the process of switching to Menu window from Sorts Bars window.

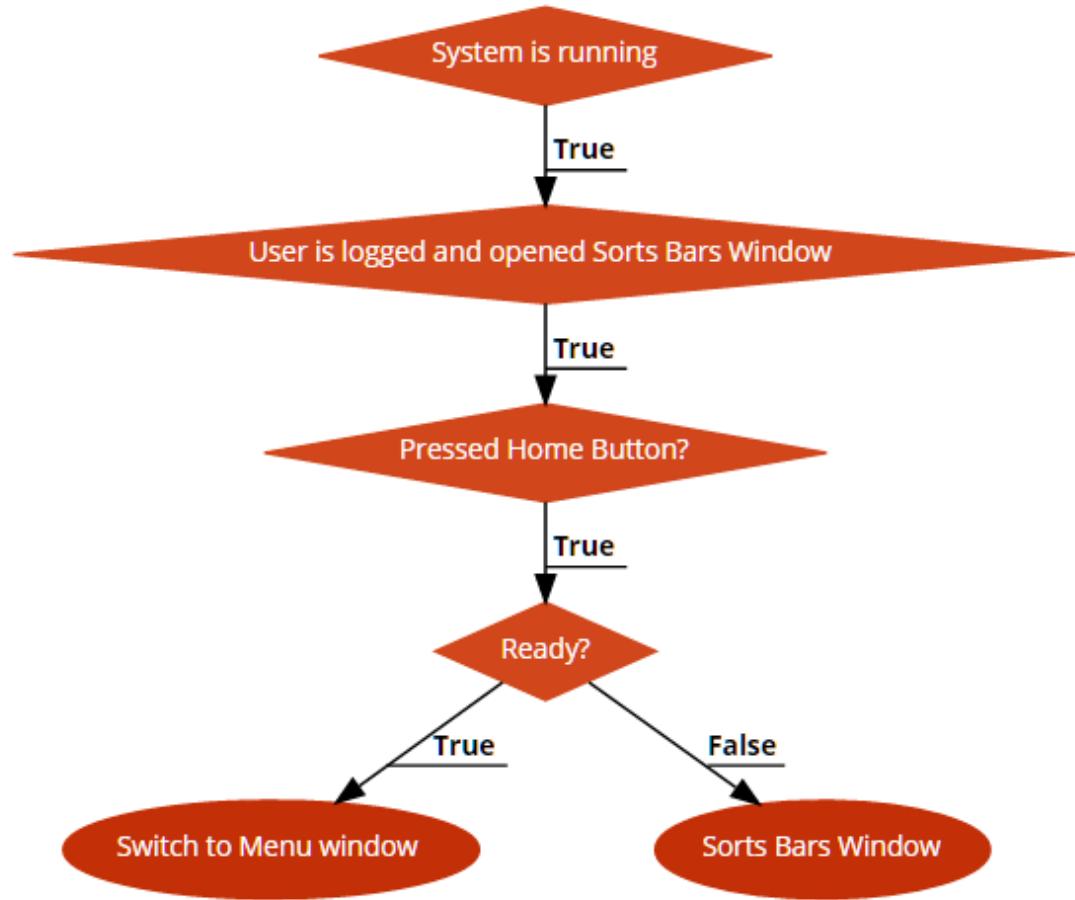


Figure 7 Flow Diagram FR7

Termination Flow Description

Precondition

User is logged in, System is running and Sorts Bars Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1.User is in Sorts Bars Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Sorts Bars window.

Termination

The user press Home button

Post condition

The system goes to Menu window.

Post condition Mock

Visual data
NCIRL Science in Computing, Software Development, Project, Jacek Byzdra, 2016 jacek

ALGORITHMS

Quiz Description

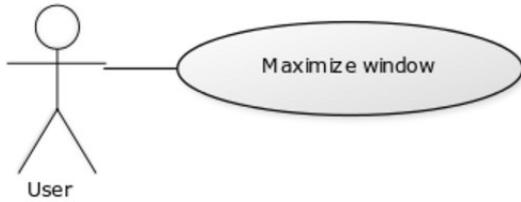
Hints Sorts Bars Sorts Panels Trees Graphs

BUBBLE SORT ALGORITHM (BSA) IS BASED ON COMPARISON OF TWO SUBSEQUENT ELEMENTS AND SWAP OF THEM EACH OTHER IF THEY ARE NOT IN QUEUE ORDER. THE TIME COMPLEXITY OF THE ALGORITHM IS AVERAGE $O(N^2)$. THE MEMORY COMPLEXITY OF BSA IS $O(1)$. THE ALGORITHM PERFORMS $N-1$ STEPS, AND IN EACH STEP IT DOES $N-K$ COMPARISON, WHERE K IS EQUAL NUMBER OF THE STEP.

4	2	3	1	0
2	4	3	1	0
2	3	4	1	0
2	3	1	4	0
2	3	1	0	4
2	1	3	0	4
2	1	0	3	4
1	2	0	3	4
1	0	2	3	4
0	1	2	3	4

```
procedure bubbleSort( A : the number of elements to sort )
n = the number of elements(A)
do
for (i = 0; i < n-1; i++) do:
if A[i] > A[i+1] then
swap(A[i], A[i+1])
end if
end for
n = n-1
while n > 1
end procedure
```

1.1.9 Requirement 8:"FR8"



1.1.9.1 Description & Priority

Maximize Window when maximize button is pressed.

1.1.9.2 Use Case

Scope

The scope of this use case is Maximize Window when user press maximize button.

Description

This use case describes the process of Maximizing window.

Termination Flow Description

Precondition

System is running. User is logged in, Register window is not opened.

Activation

This use case starts when a user press Maximize button.

Main flow

- A1.User press maximize button
- A2. Window is maximize

Alternate flow

- B1: Window size is not changed.

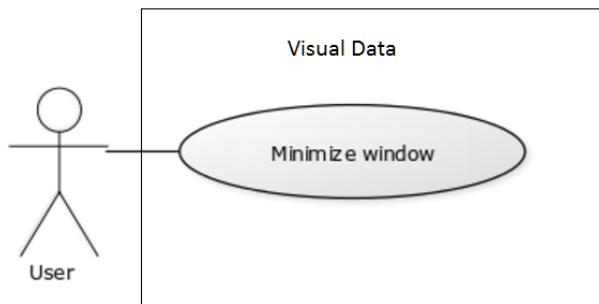
Termination

The user press Maximize button

Post condition

The system Maximize window.

1.1.10 Requirement 9:"FR9"



1.1.10.1 Description & Priority

Minimize Window when minimize button is pressed.

1.1.10.2 Use Case

Scope

The scope of this use case is Minimize Window when user press minimize button.

Description

This use case describes the process of Minimizing window.

Termination Flow Description

Precondition

System is running. User is logged in, Register window is not opened.
Window is in maximum size.

Activation

This use case starts when a user press Minimize button.

Main flow

- A1. User press minimize button
- A2. Window is minimized

Alternate flow

- B1: Window size is not changed.

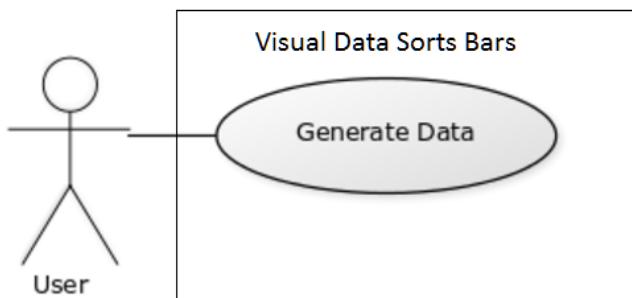
Termination

The user press Minimize button

Post condition

The system Minimize window.

1.1.11 Requirement 10:"FR10"



1.1.11.1 Description & Priority

Generate Data for sorting bars when the Generate Data button is pressed. When the program runs and Sorts Bars window is open, user provided correct integer number to sort before, and user pressed generate data button. System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label. After Generate Data button is pressed

System generates the number of bars equal to generated integer number and displays the bars in the scene.

1.1.11.2 Use Case

Scope

The scope of this use case is Generate Data when user press Generate Data Button in Sorts Bars window.

Description

This use case describes the process of Generating Data when user press Generate Data Button in Sorts Bars window.

Termination Flow Description

Precondition

System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label.

Activation

This use case starts when a user press Generate Data button.

Main flow

- A1. User press Generate Data button
- A2. System generates the number of bars equal to generated integer number and displays the bars in the scene

Alternate flow

- B1: System keeps the number of bars equal to default number 120.

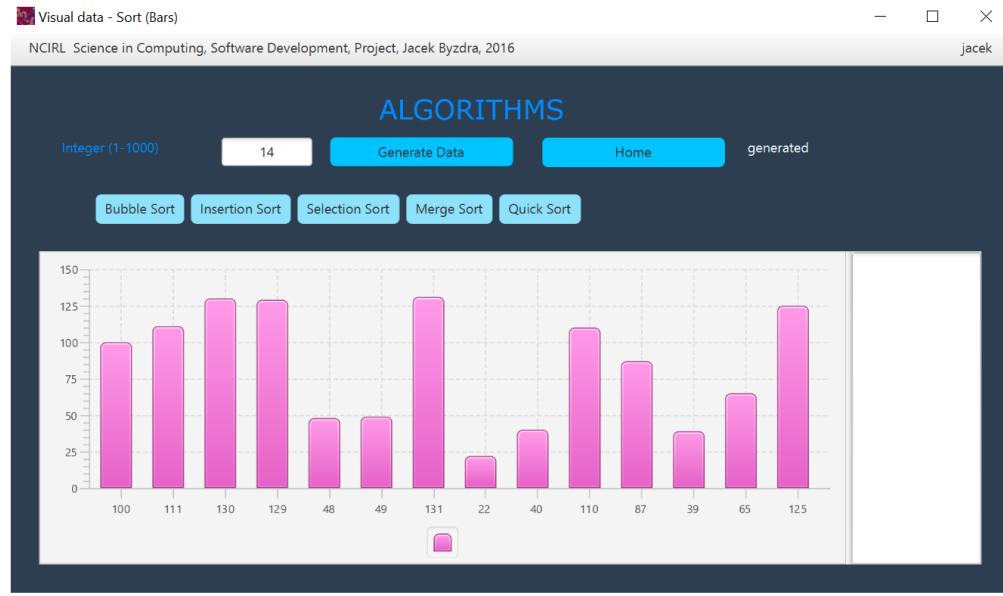
Termination

The user press Generate Data button

Post condition

System generates the number of bars equal to generated integer number and displays the bars in the scene.

Post condition Mock



1.1.12 Requirement 11:"FR11"

1.1.12.1 Description & Priority

Display information "Incorrect sort number". When the program runs and Sorts Bars window is open, user provided incorrect integer number to sort before, and user pressed generate data button. System is running, user is logged in and Sorts Bars Window is open, user provided incorrect integer number to sort label.

1.1.12.2 Use Case

Scope

The scope of this use case is Display information "Incorrect sort number". When the program runs and Sorts Bars window is open, user

provided incorrect integer number to sort before, and user pressed generate data button.

Description

This use case describes the process of displaying information "Incorrect sort number".

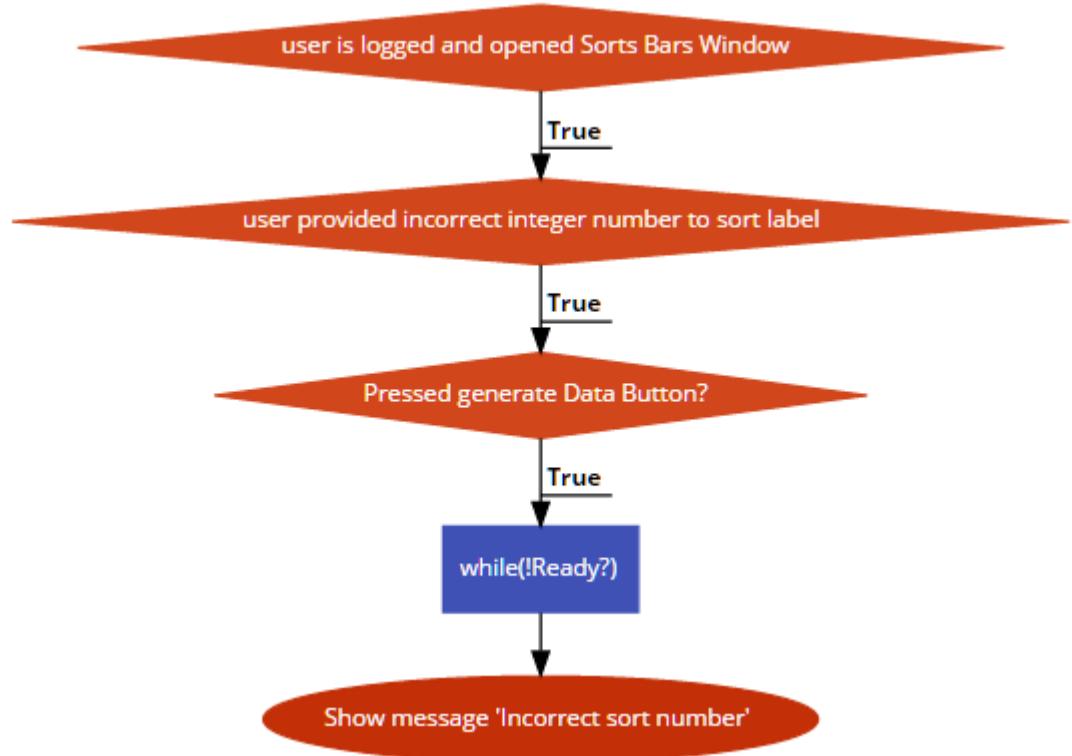


Figure 11: Flow Diagram FR 11

Termination Flow Description

Precondition

System is running, user is logged in and Sorts Bars Window is open, user provided incorrect integer number to sort label.

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

A1. User press Generate Data button

A2. System displays information "Incorrect sort number".

Alternate flow

B1: Button Generate Data is not pressed.

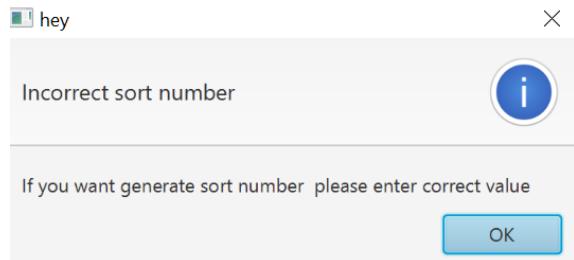
Termination

The user press Generate Data button after incorrect number was provided to sort label.

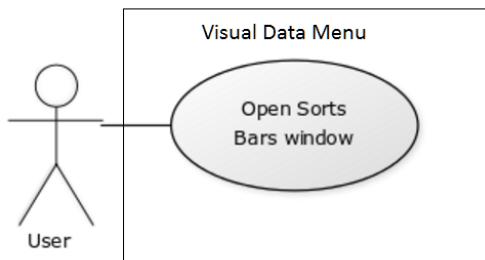
Post condition

The system displays information "Incorrect sort number".

Post condition Mock



1.1.13 Requirement 12:"FR12"



1.1.13.1 Description & Priority

Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.
When the program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

1.1.13.2 Use Case

Scope

The scope of this use case is Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window. When the program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

Description

This use case describes the process of displaying Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Termination Flow Description

Precondition

Program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

Activation

This use case starts when Sorts Bars window is opened.

Main flow

- A1. User opens Sorts Bars window
- A2. System Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Alternate flow

- B1: Sort Bars window is not opened.

Termination

The user opens Sorts Bars window.

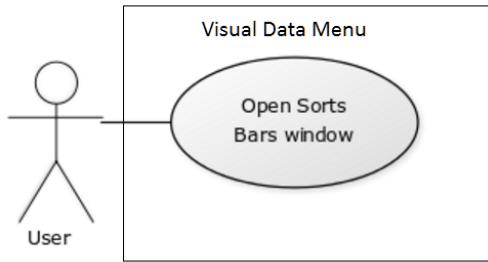
Post condition

The system Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Post condition Mock



1.1.14 Requirement 13:"FR13"



1.1.14.1 Description & Priority

Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window. When the program runs in Sorts Bars window, and user entered and generated correct integer number to sort. When the program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort.

1.1.14.2 Use Case

Scope

The scope of this use case is Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Description

This use case describes the process of displaying Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Termination Flow Description

Precondition

Program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort.

Activation

This use case starts when user generated correct integer number to sort.

Main flow

- A1. User generates correct integer number to sort
- A2. System displays button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Alternate flow

- B1: user not generates number to sort.

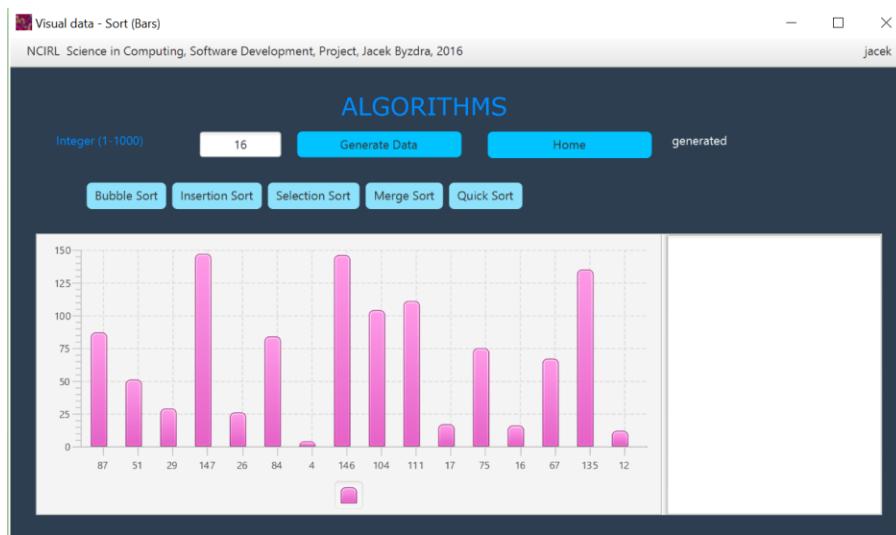
Termination

User generates correct integer number to sort.

Post condition

System displays button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Post condition Mock



1.1.15 Requirement 14:"FR14"

1.1.15.1Description & Priority

Display information about logged user in Sorts Bars window. When the program runs , user is logged in, and Sorts Bars window is opend.

1.1.15.2Use Case

Scope

The scope of this use case is Display information about logged user in Sorts Bars window. When the program runs , user is logged in, and Sorts Bars window is opend.

Description

This use case describes the process of displaying information about logged user in Sorts Bars window.

Termination Flow Description

Precondition

Program runs , user is logged in, and Sorts Bars window is opend.

Activation

This use case starts when user user is logged in, and Sorts Bars window is opend.

Main flow

- A1. After login user is directed to Sorts Bars window
- A2. System displays information about logged user in right top corner.

Alternate flow

- B1: user is not logged to system.

Termination

User successfully is logged to system.

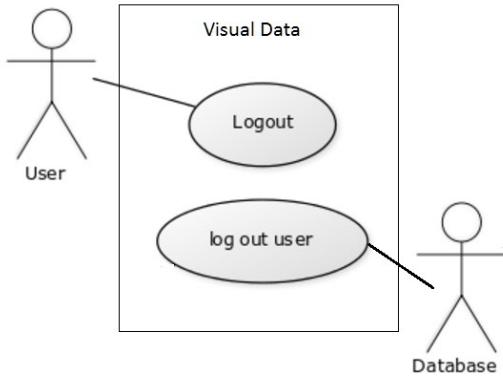
Post condition

System displays information about logged user.

Post condition Mock



1.1.16 Requirement 15:"FR15"



1.1.16.1 Description & Priority

Log user out after Logout button is pressed. When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button. When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button.

1.1.16.2 Use Case

Scope

The scope of this use case is Log user out after Logout button is pressed.

Description

This use case describes the process of logging user out after Logout button is pressed.

Termination Flow Description

Precondition

Program runs , user is logged in, and Sorts Bars window is opend.

Activation

User press Logout button.

Main flow

A1. User press logout button

A2. System logout user.

Alternate flow

B1: user is logged in.

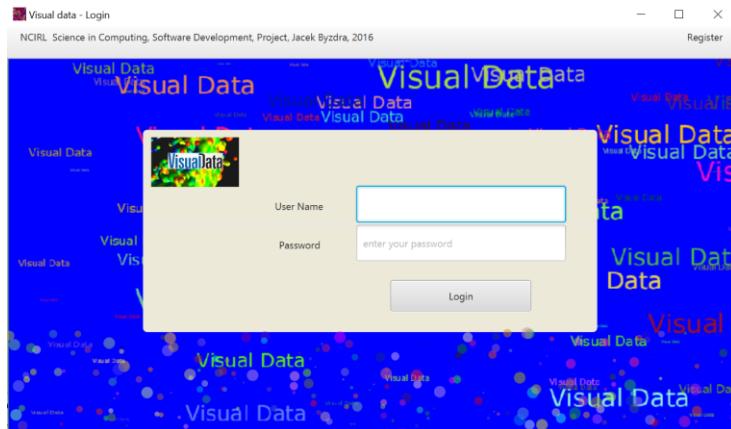
Termination

User successfully is logged out from system.

Post condition

System switch user to Login window.

Post condition Mock



1.1.17 Requirement 16:"FR16"

1.1.17.1 Description & Priority

Visual Animation of Bubble Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user

press Bubble Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

1.1.17.2 Use Case

Scope

The scope of this use case is Visual Animation of Bubble Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Bubble Sort Algorithm (Sorts Panels View).

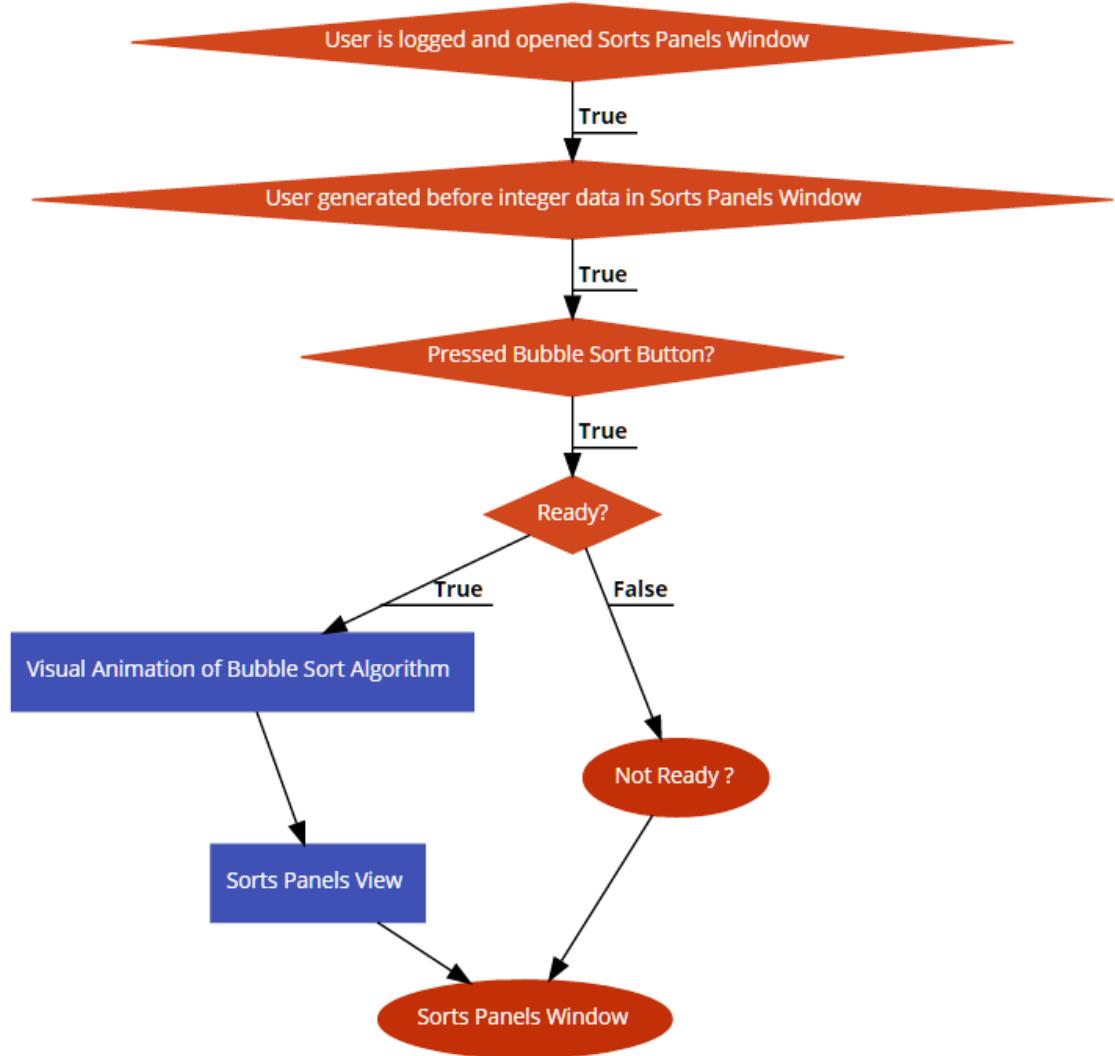


Figure 16: Flow Diagram FR16

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Bubble Sort button.

Main flow

- A1. User press Bubble sort button
- A2. System animates Bubble Sort algorithm.

Alternate flow

- B1: Bubble Sort button is not pressed.

Termination

User press Bubble sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

1.1.18 Requirement 17:"FR17"

1.1.18.1 Description & Priority

Visual Animation of Insertion Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Insertion Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

1.1.18.2 Use Case

Scope

The scope of this use case is Visual Animation of Insertion Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Insertion Sort Algorithm (Sorts Panels View).

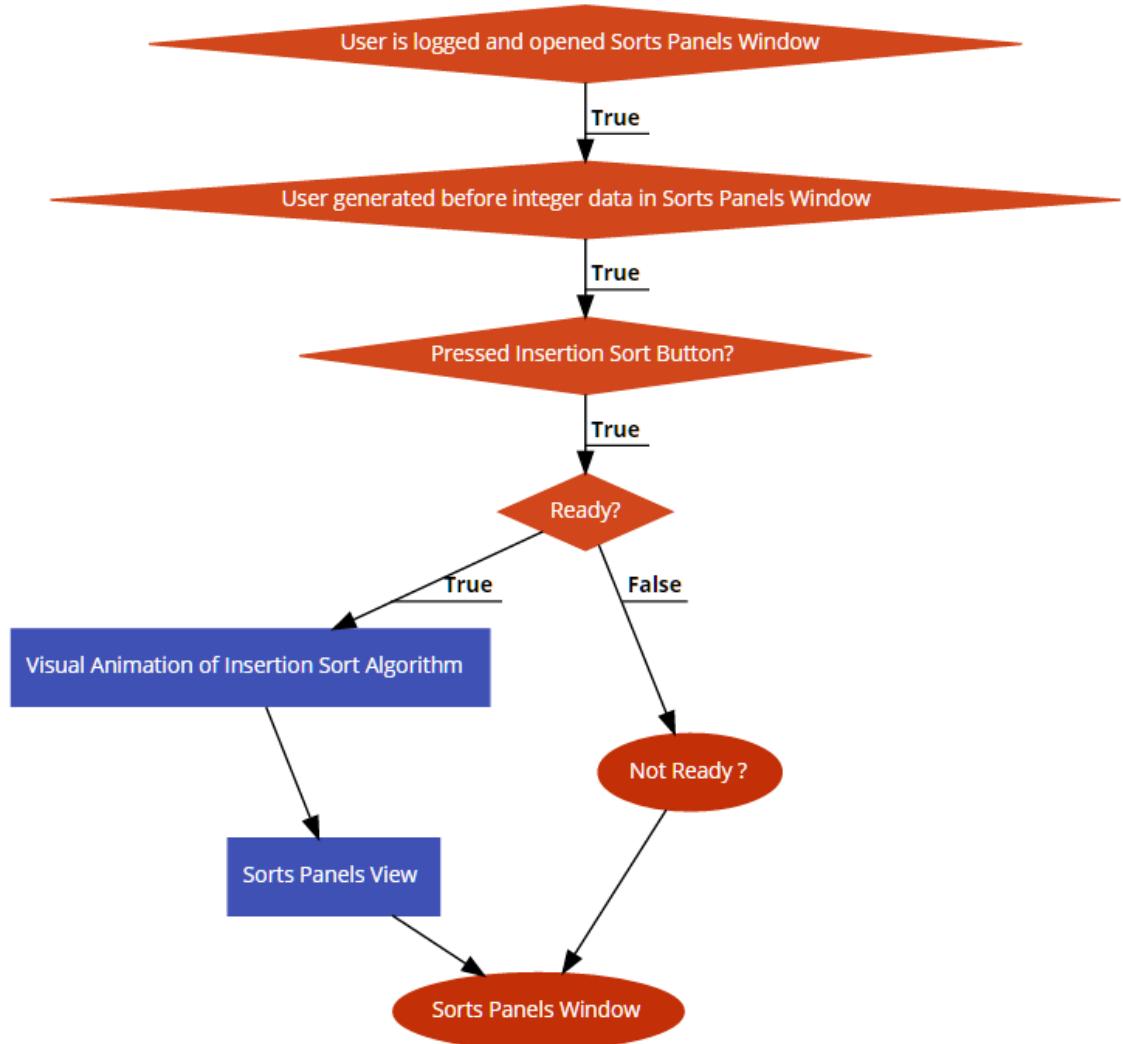


Figure 17: Flow Diagram FR17

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Insertion Sort button.

Main flow

- A1. User press Insertion sort button
- A2. System animates Insertion Sort algorithm.

Alternate flow

- B1: Insertion Sort button is not pressed.

Termination

User press Insertion sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

1.1.19 Requirement 18:"FR18"

1.1.19.1 Description & Priority

Visual Animation of Selection Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Selection Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

1.1.19.2 Use Case

Scope

The scope of this use case is Visual Animation of Selection Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Selection Sort Algorithm (Sorts Panels View).

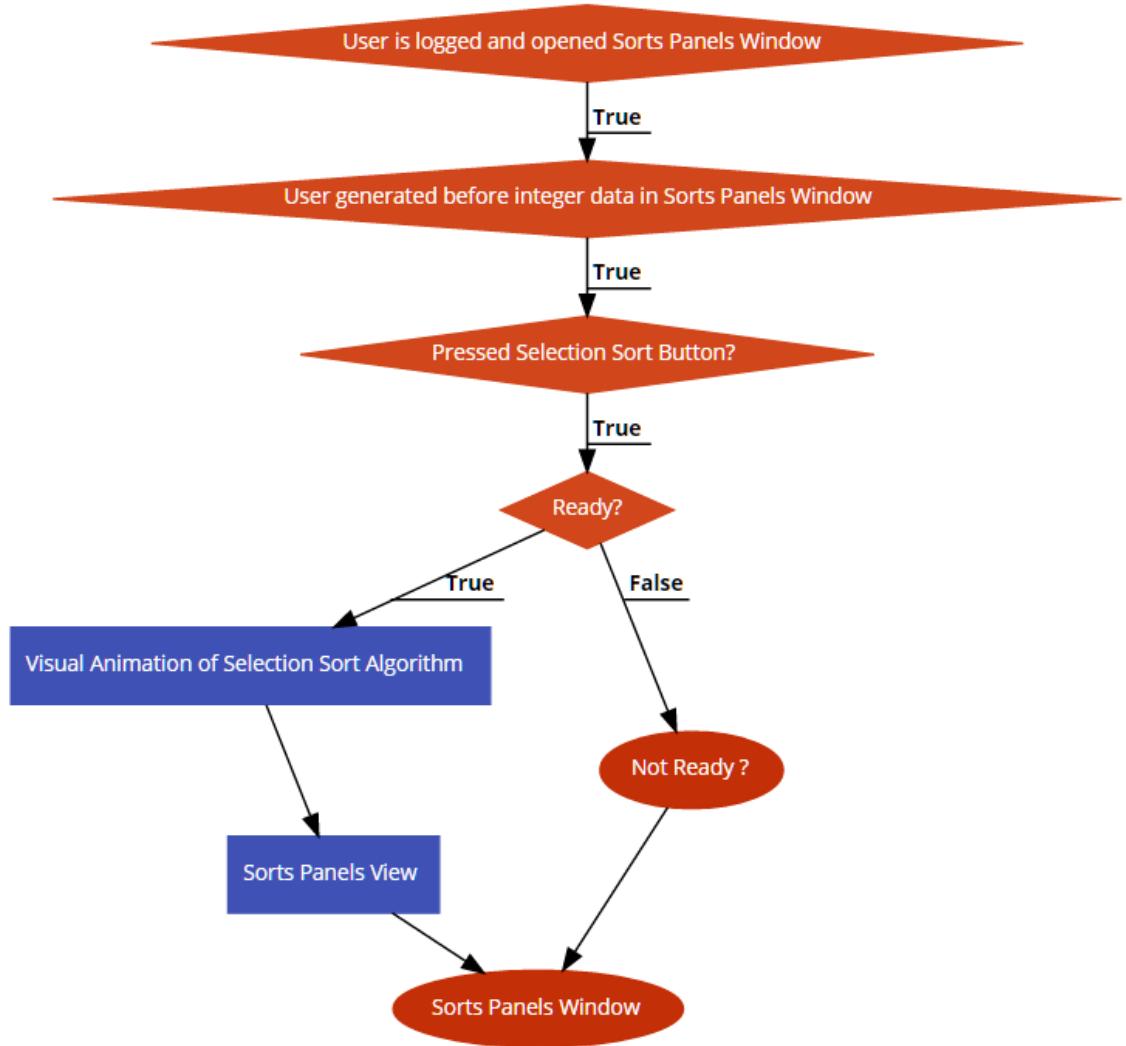


Figure 18: Flow Diagram FR18

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Selection Sort button.

Main flow

- A1. User press Selection sort button
- A2. System animates Selection Sort algorithm.

Alternate flow

- B1: Selection Sort button is not pressed.

Termination

User press Selection sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

1.1.20 Requirement 19:"FR19"

1.1.20.1 Description & Priority

Visual Animation of Heap Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Heap Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

1.1.20.2 Use Case

Scope

The scope of this use case is Visual Animation of Heap Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Heap Sort Algorithm (Sorts Panels View).

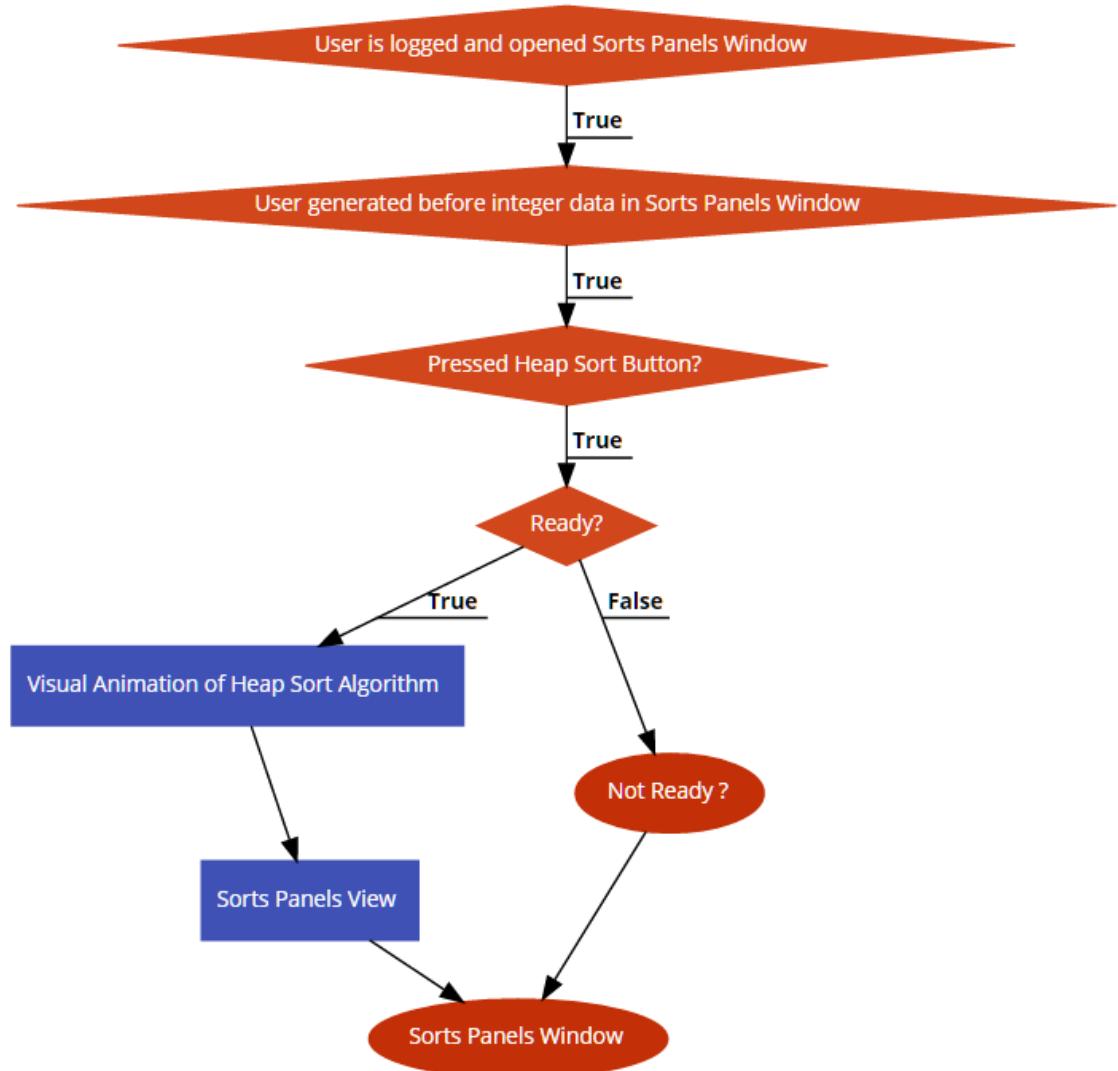


Figure 19: Flow Diagram FR19

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Heap Sort button.

Main flow

- A1. User press Heap sort button
- A2. System animates Heap Sort algorithm.

Alternate flow

B1: Heap Sort button is not pressed.

Termination

User press Heap sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

1.1.21 Requirement 20:"FR20"

1.1.21.1 Description & Priority

Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Quick Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

1.1.21.2 Use Case

Scope

The scope of this use case is Visual Animation of Quick Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Quick Sort Algorithm (Sorts Panels View).

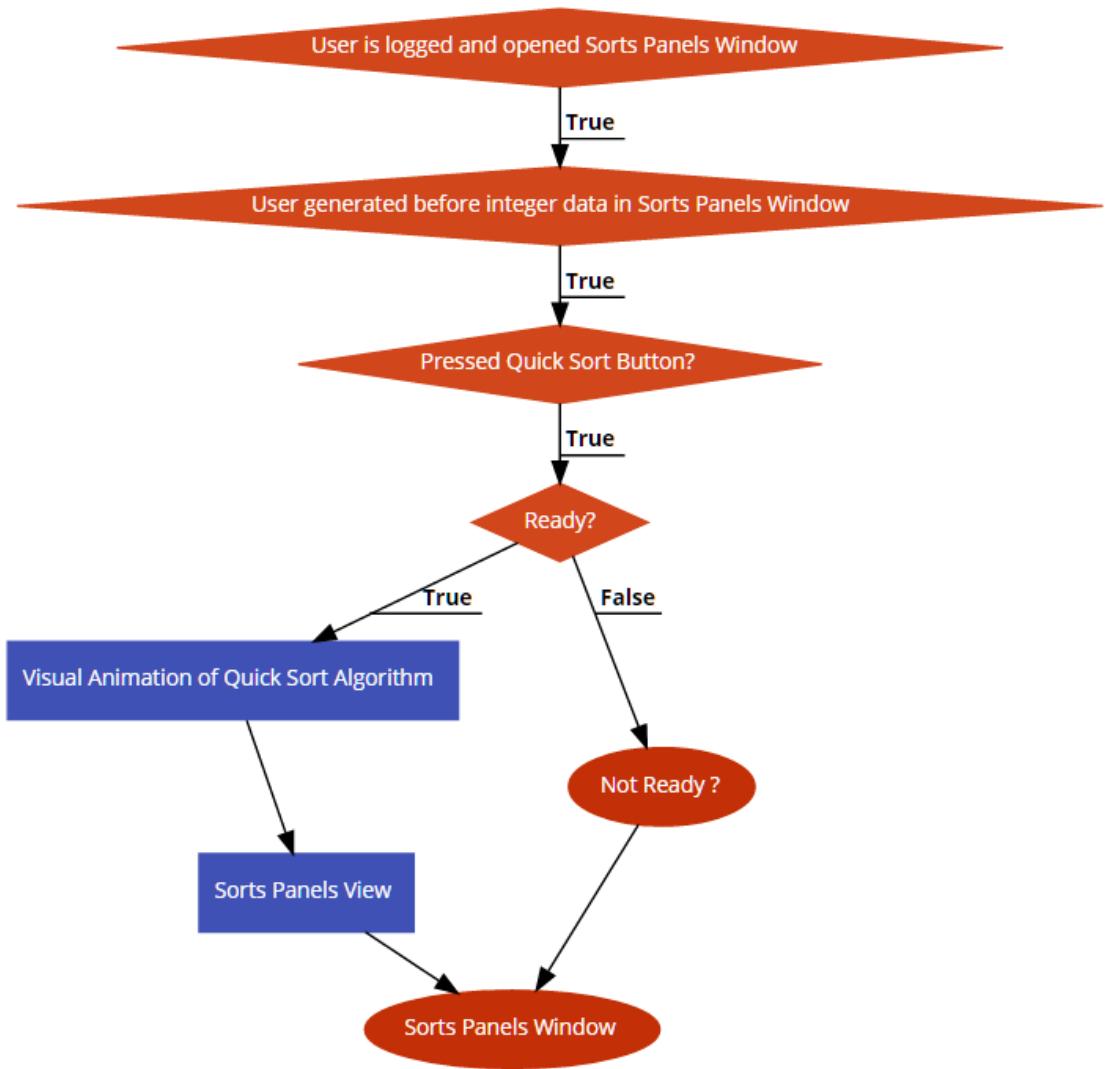


Figure 20: Flow Diagram FR20

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Quick Sort button.

Main flow

- A1. User press Quick sort button
- A2. System animates Quick Sort algorithm.

Alternate flow

B1: Quick Sort button is not pressed.

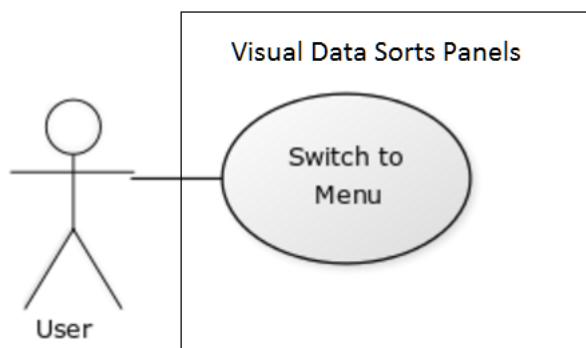
Termination

User press Quick sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

1.1.22 Requirement 21:"FR21"



1.1.22.1 Description & Priority

Switch to Menu window from Sorts Panels window after home button is pressed.
When the program runs and Sorts Panels window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Sorts Panels Window is open.

1.1.22.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Sorts Panels window.

Description

This use case describes the process of switching to Menu window from Sorts Panels window.

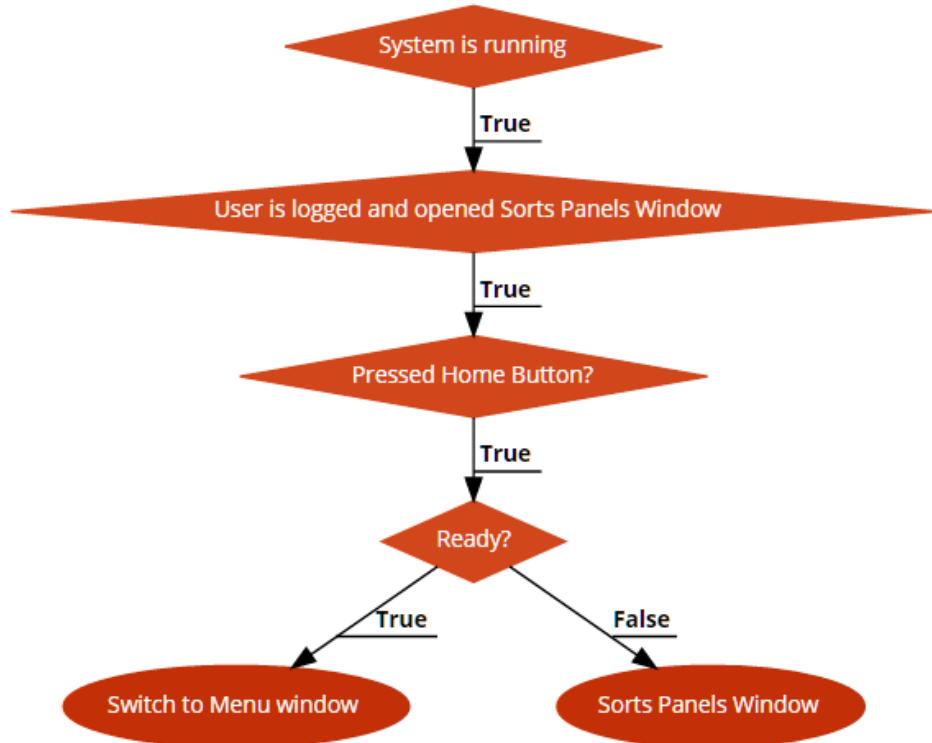


Figure 21 Flow Diagram FR21

Termination Flow Description

Precondition

User is logged in, System is running and Sorts Panels Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1.User is in Sorts Panels Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Sorts Panels window.

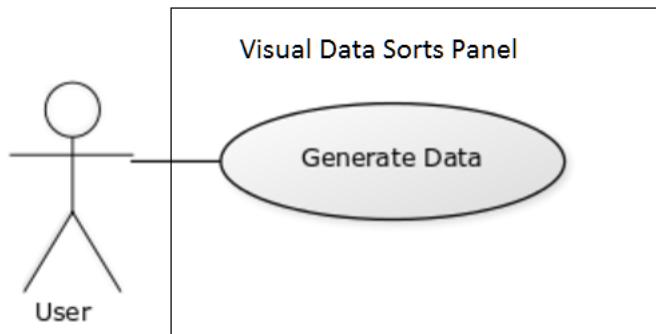
Termination

The user press Home button

Post condition

The system goes to Menu window.

1.1.23 Requirement 22:"FR22"



1.1.23.1 Description & Priority

Generate Integer Data for sorting bars when the Generate Data button is pressed. When the program runs and Sorts Panels window is open, user provided correct integer number to sort before, and user pressed generate data button. System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label.

1.1.23.2 Use Case

Scope

The scope of this use case is to Generate Integer Data for sorting bars when the Generate Data button is pressed.

Description

This use case describes the process of Generating Integer Data for sorting bars when the Generate Data button is pressed.

Figure 21 Flow Diagram FR21

Termination Flow Description

Precondition

System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label.

Activation

This use case starts when a user press Generate Data button.

Main flow

A1. User is in Sorts Panels Window and press Generate Data button.

A2. System generates and draws the number of panels equal the number defined by user

Alternate flow

B1: The system Generate Data Button is not pressed.

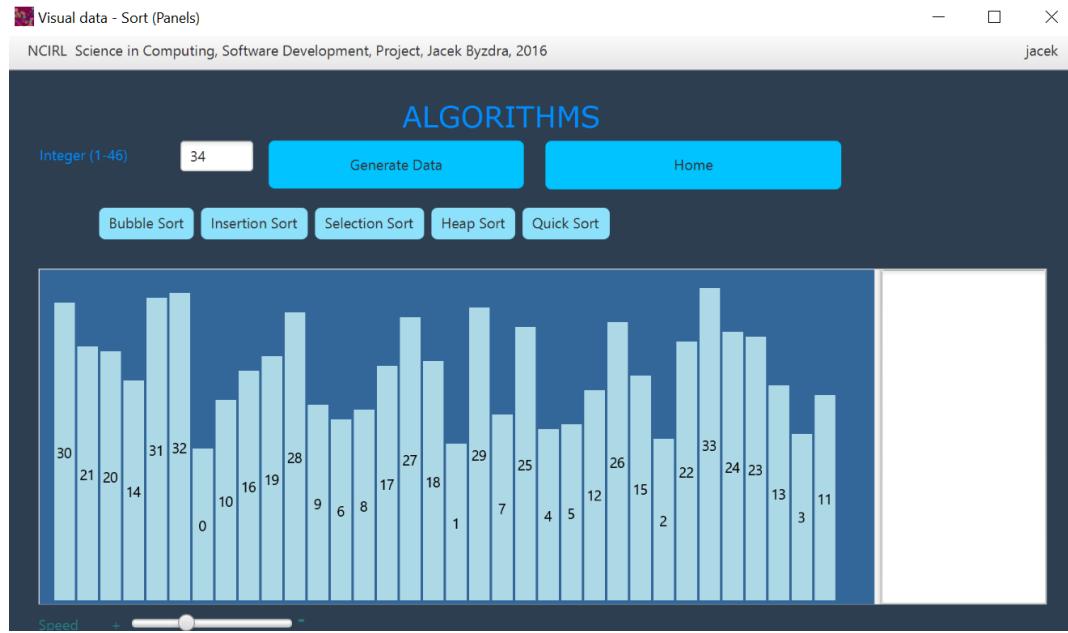
Termination

The user press Generate Data button

Post condition

The system display in the scene the number of panels equal the number defined by user.

Post Mock-up



1.1.24 Requirement 23:"FR23"

1.1.24.1 Description & Priority

Display information "Incorrect sort number". When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button. System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label.

1.1.24.2 Use Case

Scope

The scope of this use case is Display information "Incorrect sort number". When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button.

Description

This use case describes the process of displaying information "Incorrect sort number".

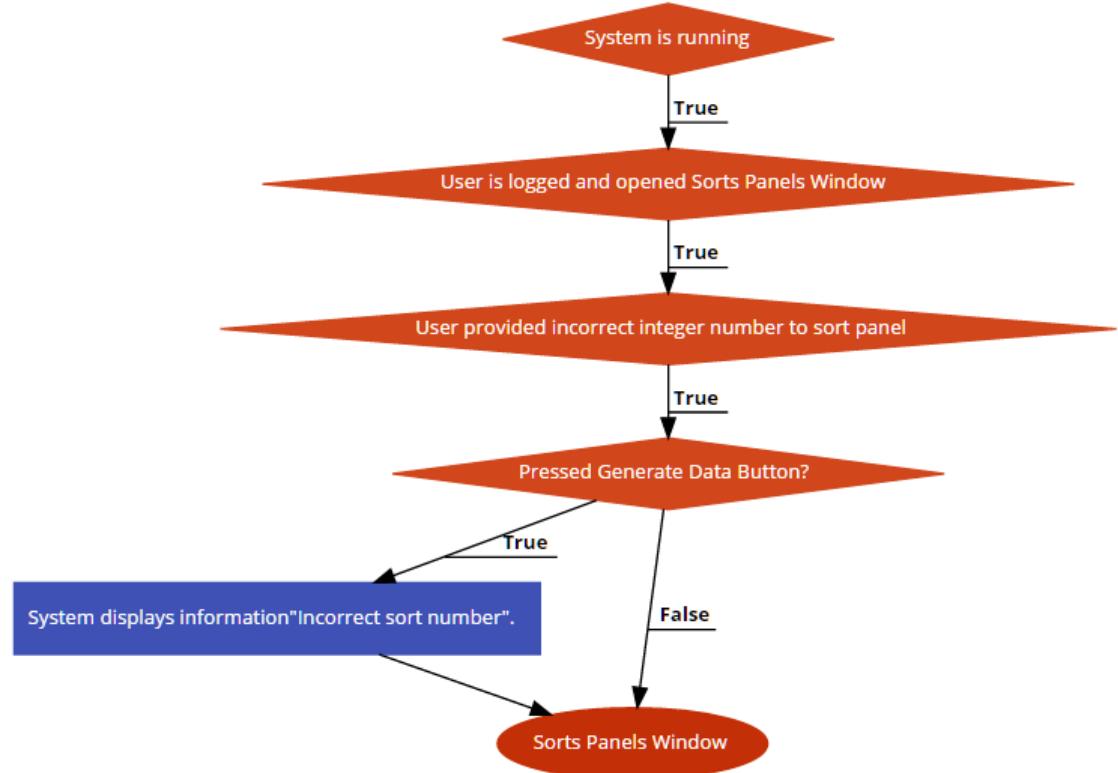


Figure 23: Flow Diagram FR 23

Termination Flow Description

Precondition

System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label.

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

- A1. User press Generate Data button
- A2. System displays information "Incorrect sort number".

Alternate flow

B1: Button Generate Data is not pressed.

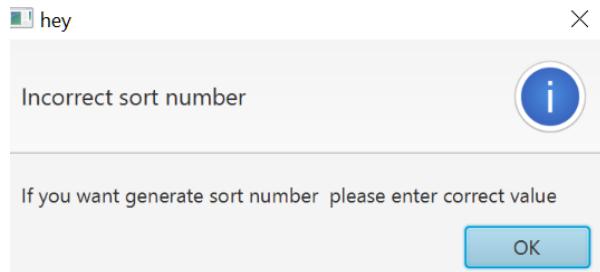
Termination

The user press Generate Data button after incorrect number was provided to sort label.

Post condition

The system displays information "Incorrect sort number".

Post condition Mock



1.1.25 Requirement 40:"FR40"

1.1.25.1 Description & Priority

Display information "Enter correct number of nodes". When the program runs in Binary Search Tree window, user provided before incorrected number of nodes to generate, and pressed Generate Data button. System is running, user is logged in, Binary Search Tree window is opened , user provided incorrect number to BST search label.

1.1.25.2 Use Case

Scope

The scope of this use case is Display information "Enter correct number of nodes". When the program runs in Binary Search Tree window, user

provided before incorrected number of nodes to generate, and pressed Generate Data button.

Description

This use case describes the process of displaying information "Enter correct number of nodes".

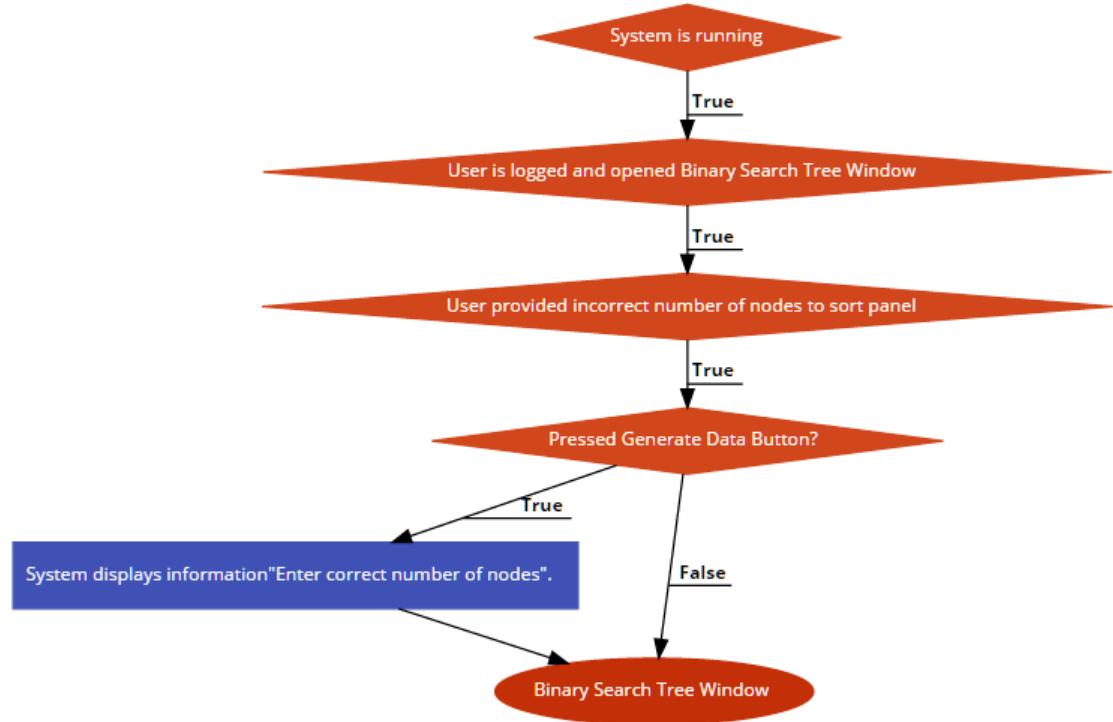


Figure 40: Flow Diagram FR 40

Termination Flow Description

Precondition

System is running, user is logged in, Binary Search Tree window is opened, user provided incorrect number to BST search label .

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

A1. User press Generate Data button

A2. System displays information "Enter correct number of nodes".

Alternate flow

B1: Button Generate Data is not pressed.

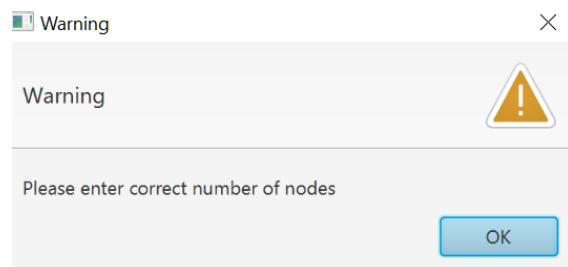
Termination

The user press Generate Data button after provided incorrected number of nodes to sort label.

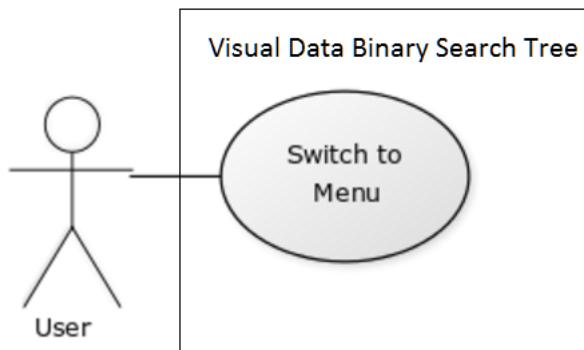
Post condition

The system displays information " Enter correct number of nodes ".

Post condition Mock



1.1.26 Requirement 44:"FR44"



1.1.26.1 Description & Priority

Switch to Menu window from Binary Search Tree window after home button is pressed. When the program runs and Binary Search Tree window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Binary Search Tree Window is open.

1.1.26.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Binary Search Tree window.

Description

This use case describes the process of switching to Menu window from Binary Search Tree window.

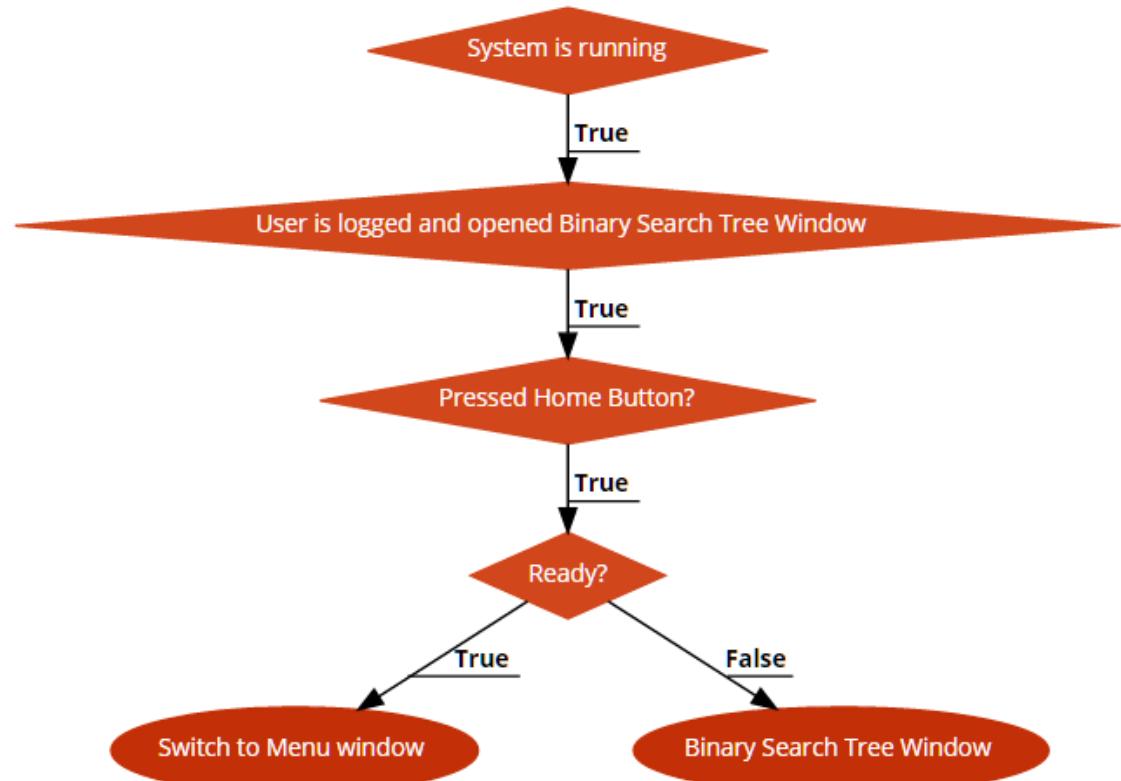


Figure 44 Flow Diagram FR44

Termination Flow Description

Precondition

User is logged in, System is running and Binary Search Tree Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1. User is in Binary Search Tree Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Binary Search Tree window.

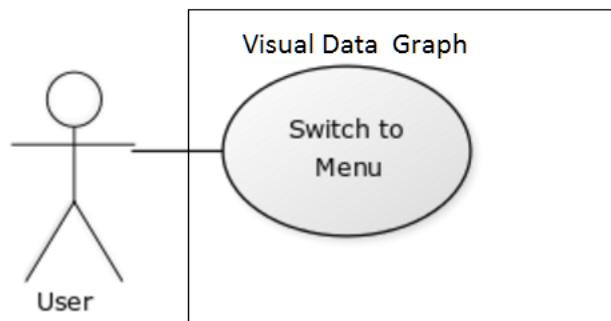
Termination

The user press Home button

Post condition

The system goes to Menu window.

1.1.27 Requirement 46:"FR46"



1.1.27.1 Description & Priority

Switch to Menu window from Graphs window after home button is pressed. When the program runs and Graphs window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Graphs Window is open.

1.1.27.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Graphs window.

Description

This use case describes the process of switching to Menu window from Graphs window.

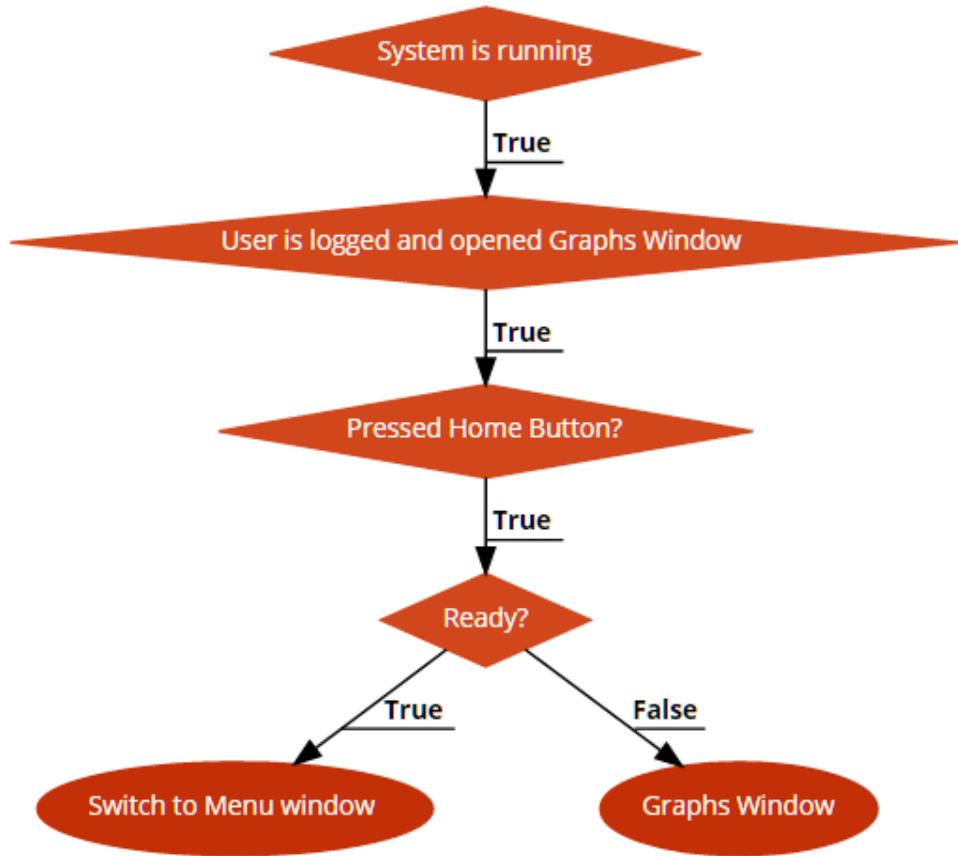


Figure 46 Flow Diagram FR46

Termination Flow Description

Precondition

User is logged in, System is running and Graphs Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1. User is in Graphs Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Graphs window.

Termination

The user press Home button

Post condition

The system goes to Menu window.

1.1.28 Requirement 48:"FR48"

1.1.28.1 Description & Priority

Display information "Incorrect graph size If you want generate graph please enter correct graph size" When the program runs and Graph window is open, user provided incorrect integer number to create graph before, and user pressed Generate Graph button. System is running, user is logged, Graph Window is open, user provided incorrect integer number to graph label.

1.1.28.2 Use Case

Scope

The scope of this use case is Display information information "Incorrect graph size If you want generate graph please enter correct graph size". When the program runs and Graph window is open, user provided incorrect integer number to create graph before, and user pressed generate graph button.

Description

"Incorrect graph size If you want generate graph please enter correct graph size".

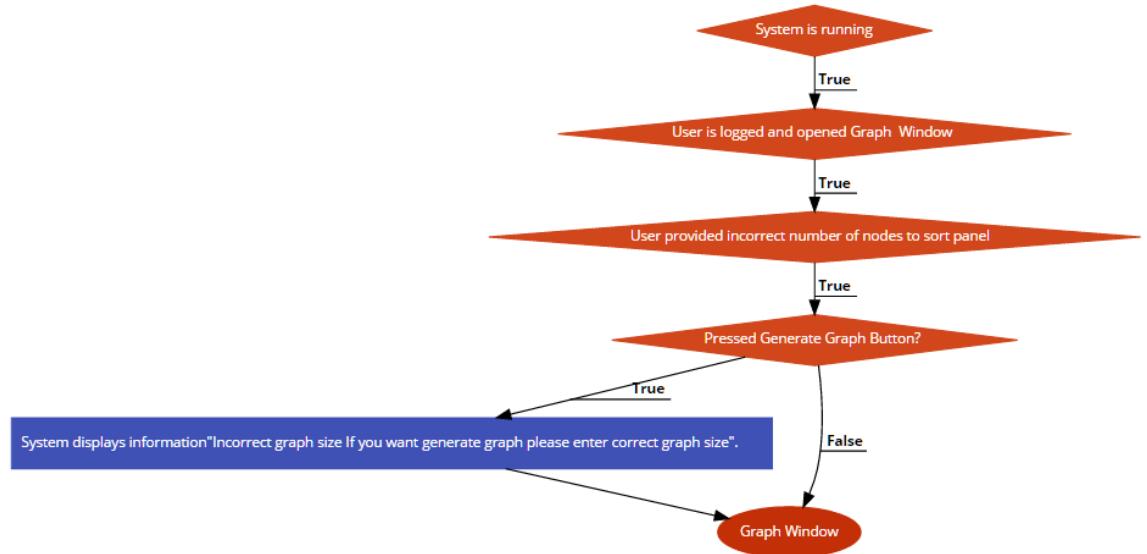


Figure 48: Flow Diagram FR 48

Termination Flow Description

Precondition

System is running, user is logged, Graph Window is open, user provided incorrect integer number to graph label.

Activation

This use case starts when a user presses Generate Graph button after an incorrect number was provided to graph label.

Main flow

- A1. User presses Generate Graph button
- A2. System displays information "Incorrect graph size If you want generate graph please enter correct graph size".

Alternate flow

- B1: Generate Graph button is not pressed.

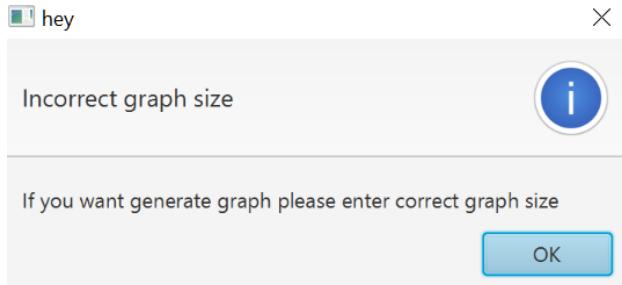
Termination

The user presses Generate Graph button after providing an incorrect number of nodes to graph label.

Post condition

The system displays information "Incorrect graph size If you want generate graph please enter correct graph size".

Post condition Mock



1.1.29 Requirement 55:"FR55"

1.1.29.1 Description & Priority

Display information: please enter correct index for start node and for end node and click Set button, when the provided value of start end node was incorrect. When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button. When the program runs, user is logged in, and Graph window is opened, and user provided incorrect number of nodes to create graph, and press buttonet start/end node in graph.

1.1.29.2 Use Case

Scope

The scope of this use case is Display information "Please enter correct index for start node and for end node and click Set button". When the provided value of start end node was incorrect. When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button.

Description

"Incorrect graph size If you want generate graph please enter correct graph size".

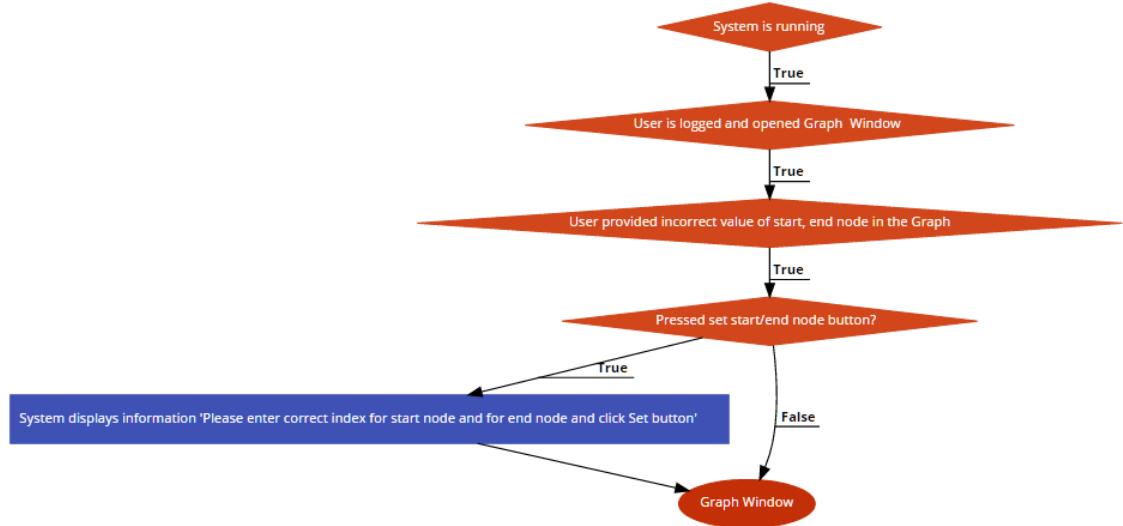


Figure 55: Flow Diagram FR 55

Termination Flow Description

Precondition

System is running, user is logged, Graph Window is open, and user provided incorrect number of nodes to create graph, and press button start/end node in graph.

Activation

This use case starts when a user pressed set start/end node button after provided value of start, end node in the Graph was incorrect.

Main flow

- A1. User press set start/end node button
- A2. System displays information " Incorrect graph size If you want generate graph please enter correct graph size ".

Alternate flow

- B1: start/end node in button in graphs window is not pressed.

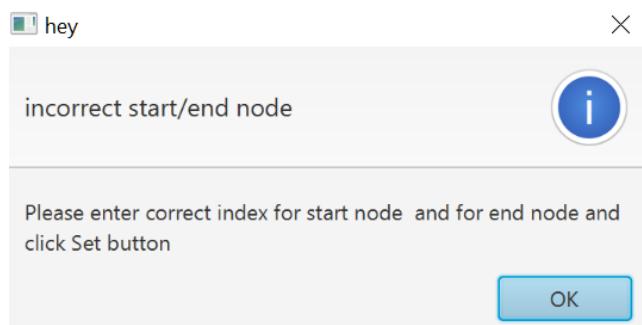
Termination

The user press start/end node button after provided incorrect value of start, end node in the Graph .

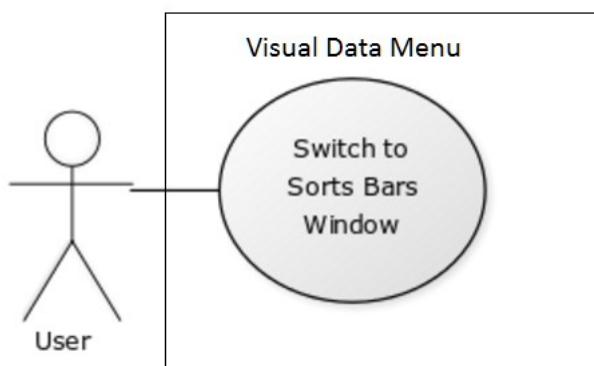
Post condition

The system displays information "Please enter correct index for start node and for end node and click Set button ".

Post condition Mock



1.1.30 Requirement 60:"FR60"



1.1.30.1 Description & Priority

Switch to Sorts Bars window from Menu Window after Sorts Bars button is pressed. When the program runs and Menu window is opened. System should switch to Sorts Bars window after Sorts Bars button is pressed. User is logged in, Menu window is opened, System is running.

1.1.30.2 Use Case

Scope

The scope of this use case is to Switch to Sorts Bars window from Menu window.

Description

This use case describes the process of switching to Sorts Bars window from Menu window.

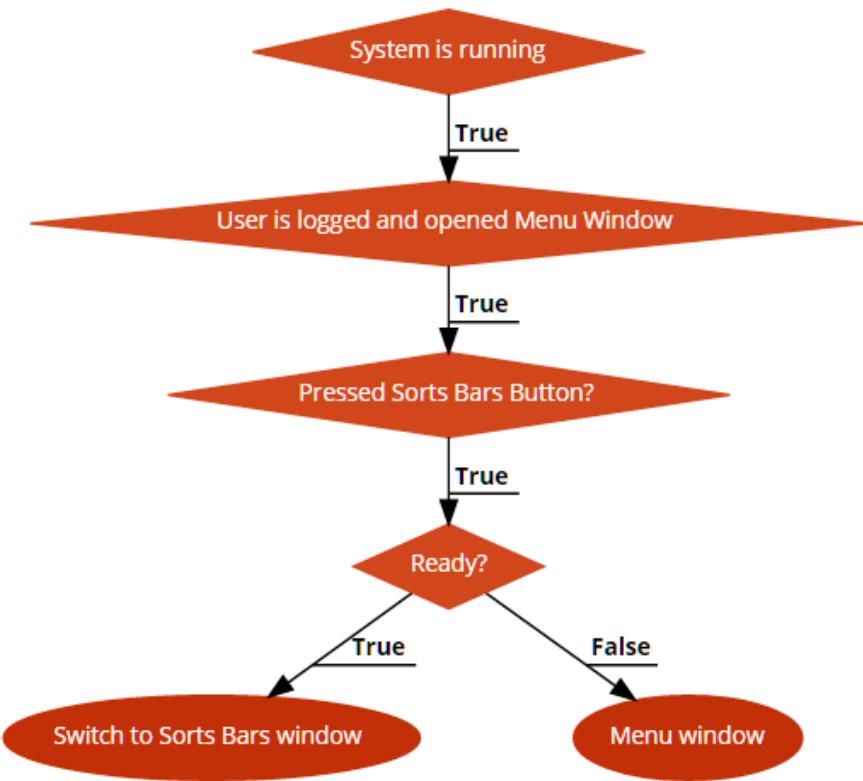


Figure 60 Flow Diagram FR60

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Sorts Bars button.

Main flow

A1. User is in Menu Window and press Sorts Bars button.

A2. System switch to Sorts Bars window

Alternate flow

B1: The system shows Menu window.

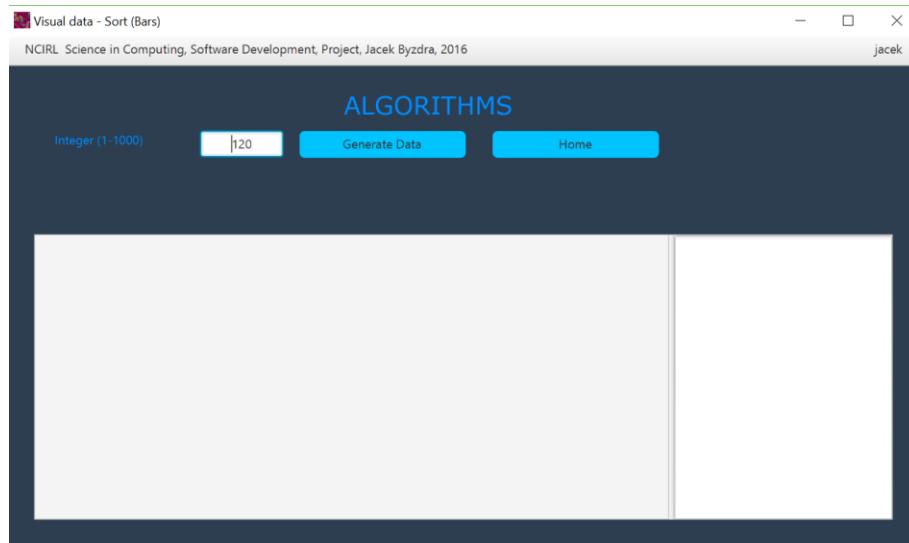
Termination

The user press Sorts Bars button

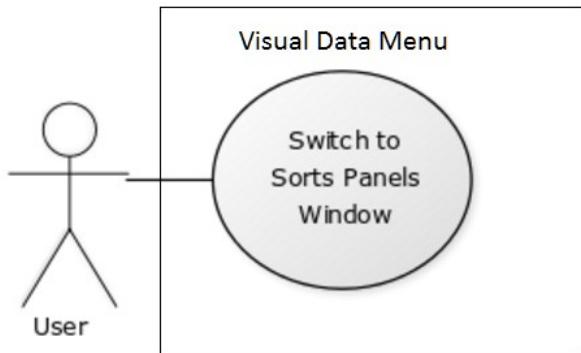
Post condition

The system goes to Sorts Bars window.

Post Mock-up



1.1.31 Requirement 61:"FR61"



1.1.31.1 Description & Priority

Switch to Sorts Panels window from Menu Window after Sorts Panels button is pressed. When the program runs and Menu window is opened. System should switch to Sorts Panels window after Sorts Panels button is pressed. User is logged in, Menu window is opened, System is running.

1.1.31.2 Use Case

Scope

The scope of this use case is to Switch to Sorts Panels window from Menu window.

Description

This use case describes the process of switching to Sorts Panels window from Menu window.

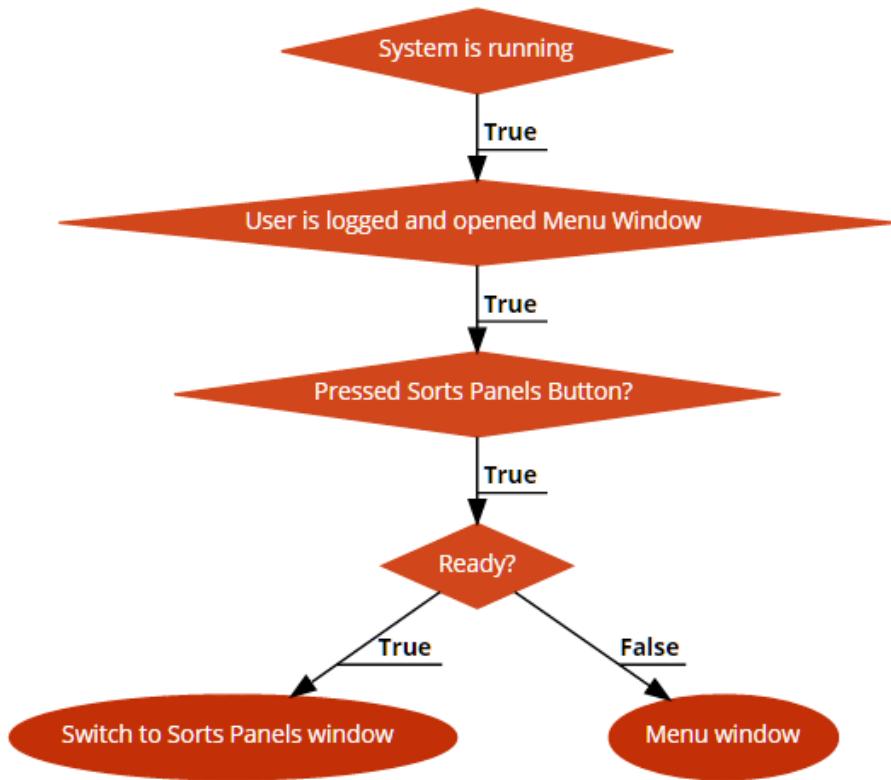


Figure 61 Flow Diagram FR61

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Sorts Panels button.

Main flow

A1. User is in Menu Window and press Sorts Panels button.

A2. System switch to Sorts Panels window

Alternate flow

B1: The system shows Menu window.

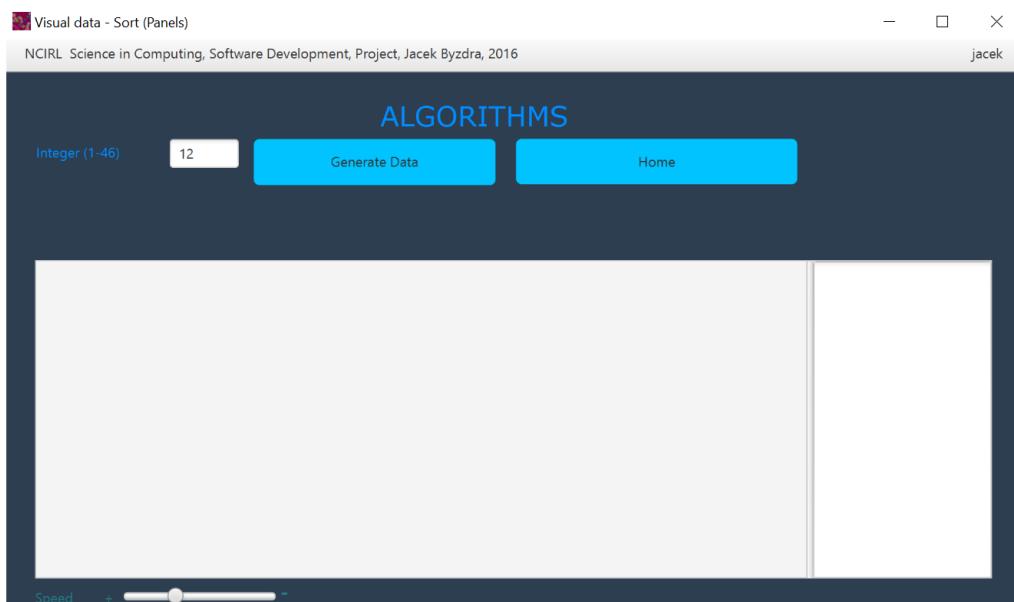
Termination

The user press Sorts Panels button

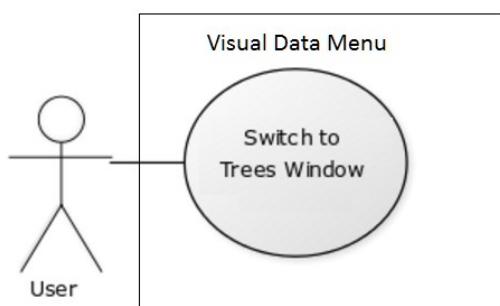
Post condition

The system goes to Sorts Panels window.

Post Mock-up



1.1.32 Requirement 62:"FR62"



1.1.32.1 Description & Priority

Switch to Trees window from Menu Window after Trees button is pressed. When the program runs and Menu window is opened. System should switch to Trees

window after Trees button is pressed. User is logged in, Menu window is opened, System is running.

1.1.32.2 Use Case

Scope

The scope of this use case is to Switch to Trees window from Menu window.

Description

This use case describes the process of switching to Trees window from Menu window.

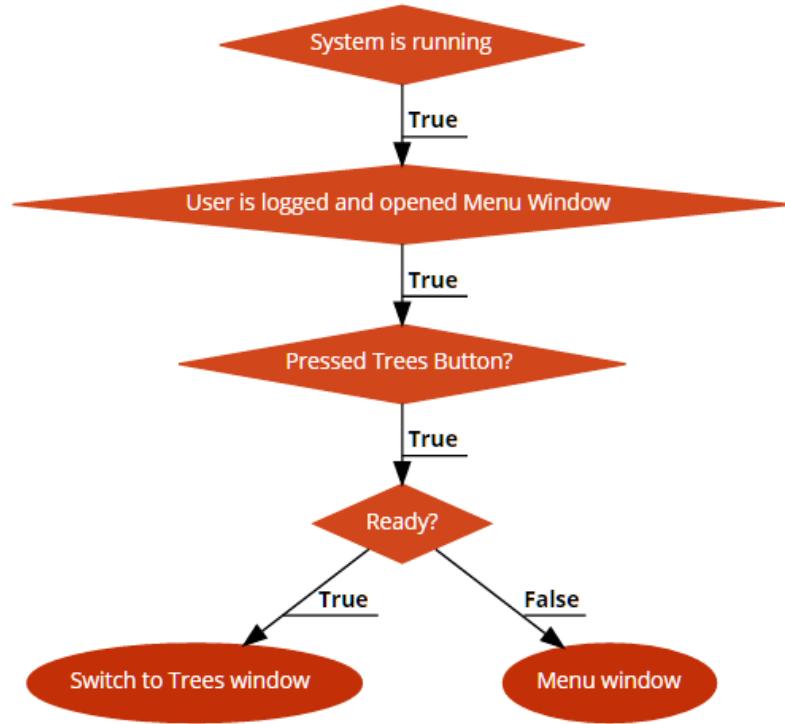


Figure 62 Flow Diagram FR62

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Trees button.

Main flow

A1. User is in Menu Window and press Trees button.

A2. System switch to Trees window

Alternate flow

B1: The system shows Menu window.

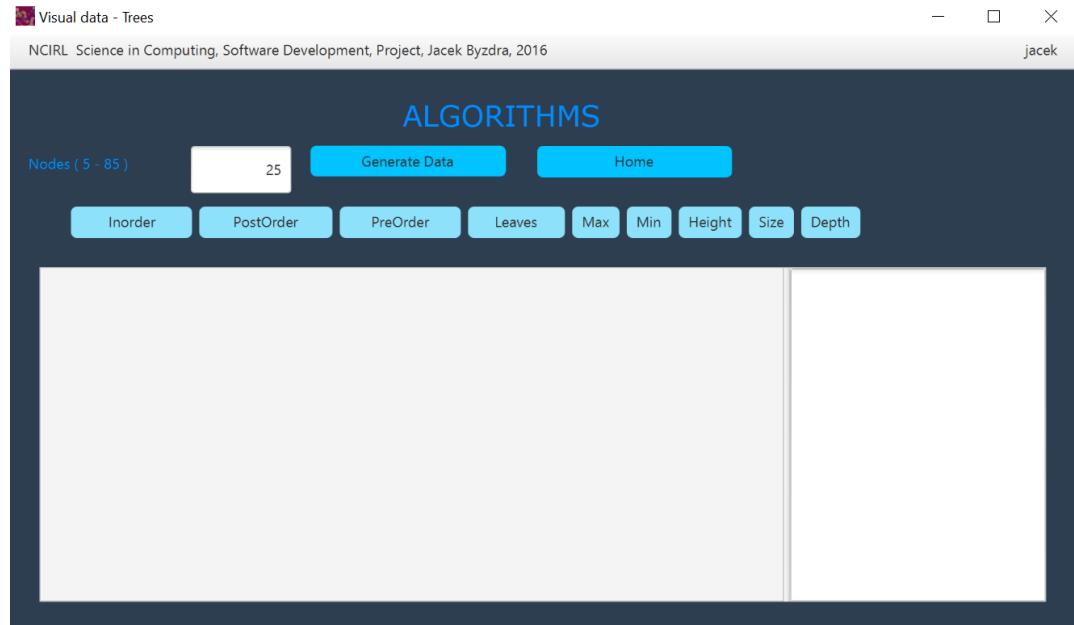
Termination

The user press Trees button

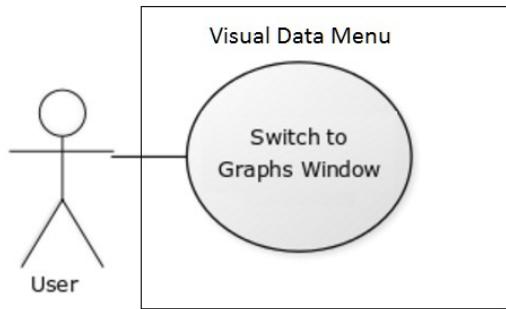
Post condition

The system goes to Trees window.

Post Mock-up



1.1.33 Requirement 63:"FR63"



1.1.33.1 Description & Priority

Switch to Graphs window from Menu Window after Graphs button is pressed. When the program runs and Menu window is opened. System should switch to Graphs window after Graphs button is pressed. User is logged in, Menu window is opened, System is running.

1.1.33.2 Use Case

Scope

The scope of this use case is to Switch to Graphs window from Menu window.

Description

This use case describes the process of switching to Graphs window from Menu window.

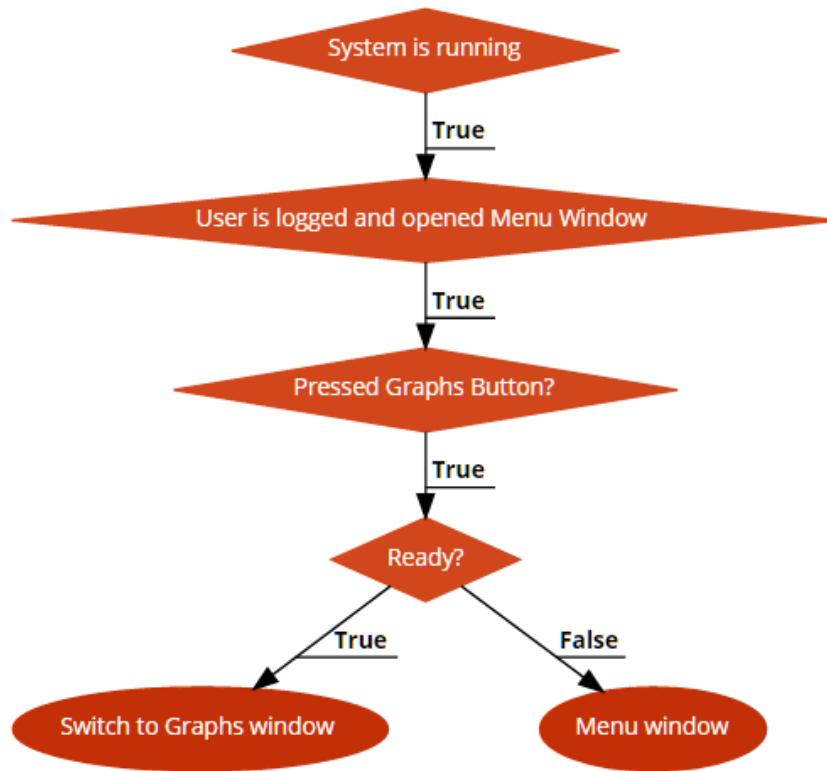


Figure 63 Flow Diagram FR63

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Graphs button.

Main flow

- A1. User is in Menu Window and press Graphs button.
- A2. System switch to Graphs window

Alternate flow

- B1: The system shows Menu window.

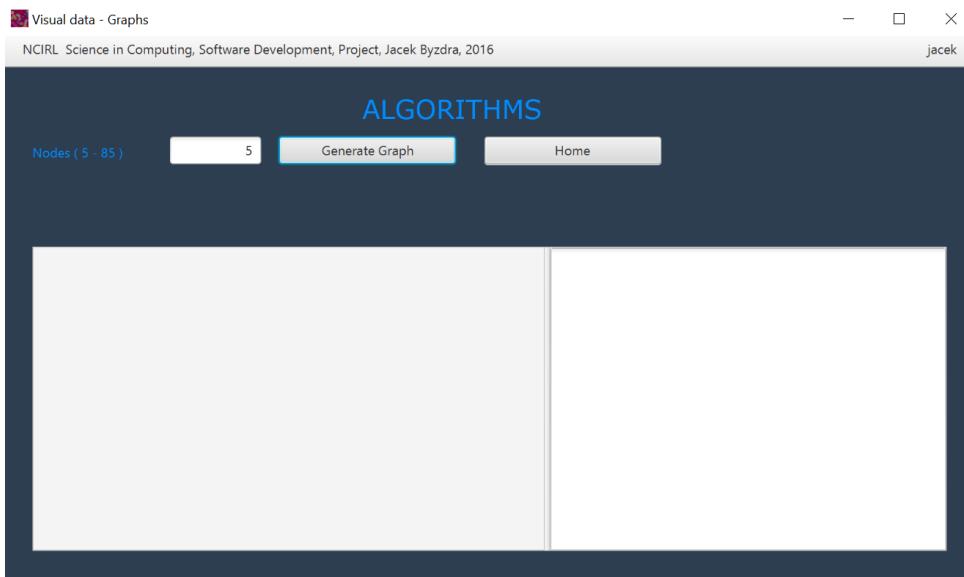
Termination

The user press Graphs button

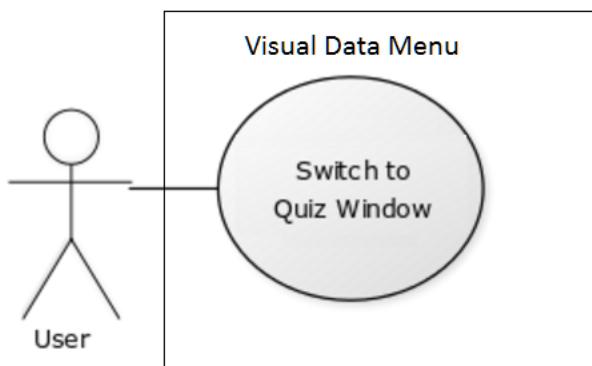
Post condition

The system goes to Graphs window.

Post Mock-up



1.1.34 Requirement 64:"FR64"



1.1.34.1 Description & Priority

Switch to Quiz window from Menu Window after Quiz button is pressed. When the program runs and Menu window is opened. System should switch to Quiz

window after Quiz button is pressed. User is logged in, Menu window is opened, System is running.

1.1.34.2 Use Case

Scope

The scope of this use case is to Switch to Quiz window from Menu window.

Description

This use case describes the process of switching to Quiz window from Menu window.

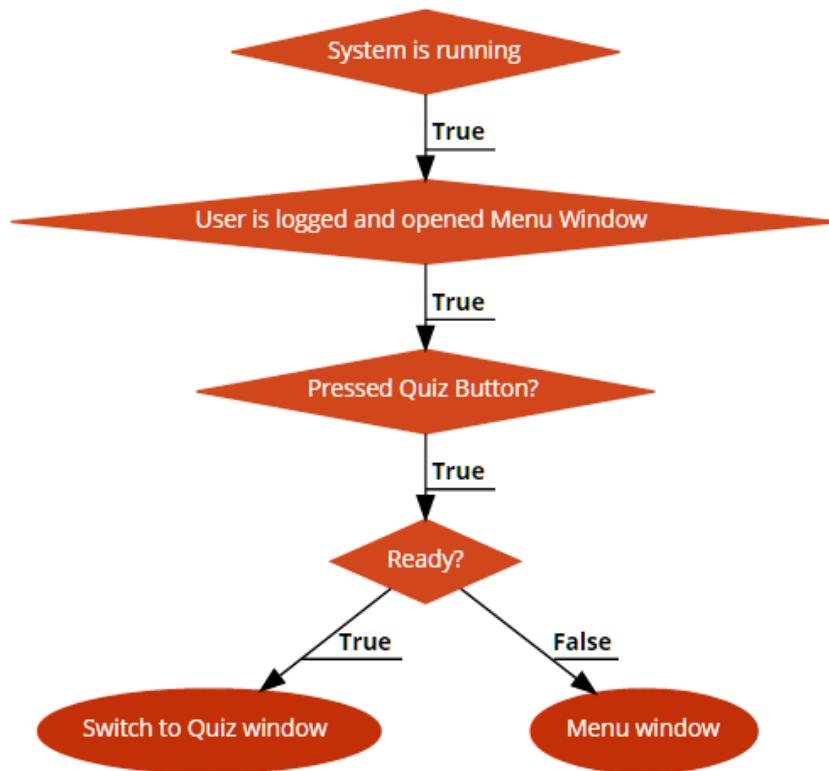


Figure 64 Flow Diagram FR64

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Quiz button.

Main flow

A1. User is in Menu Window and press Quiz button.

A2. System switch to Quiz window

Alternate flow

B1: The system shows Menu window.

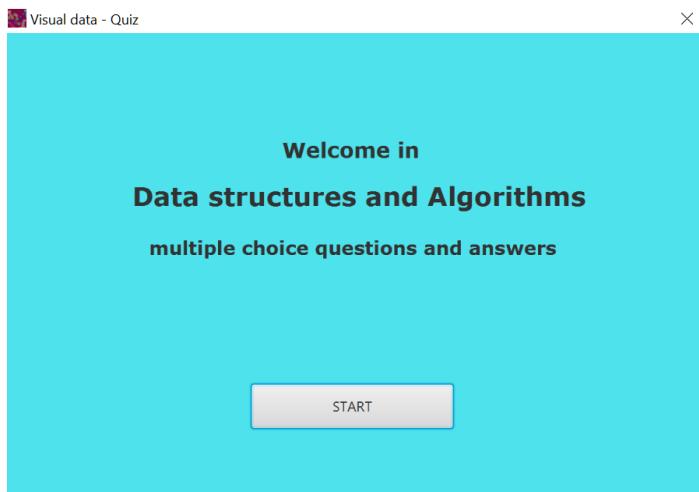
Termination

The user press Quiz button

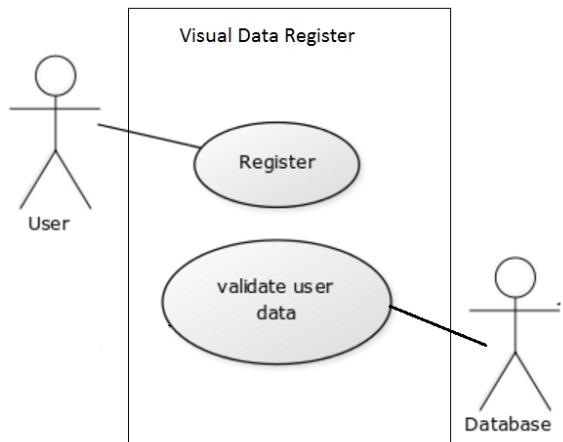
Post condition

The system goes to Quiz window.

Post Mock-up



1.1.35 Requirement 79:"FR79"



1.1.35.1 Description & Priority

Display information: 'User name is incorrect. Please enter correct user name for register'. When the user name was not provided in registration form in Registration Window and Sign Up button was pressed. When the program runs in Register Window and user not provided user name and pressed Sign Up button. Program runs in Register Window.

1.1.35.2 Use Case

Scope

The scope of this use case is Display information 'User name is incorrect. Please enter correct user name for register'. When the user name was not provided in registration form in Registration Window and Sign Up button was pressed. When the program runs in Register Window and user not provided user name and pressed Sign Up button. Program runs in Register Window..

Description

'User name is incorrect. Please enter correct user name for register'.

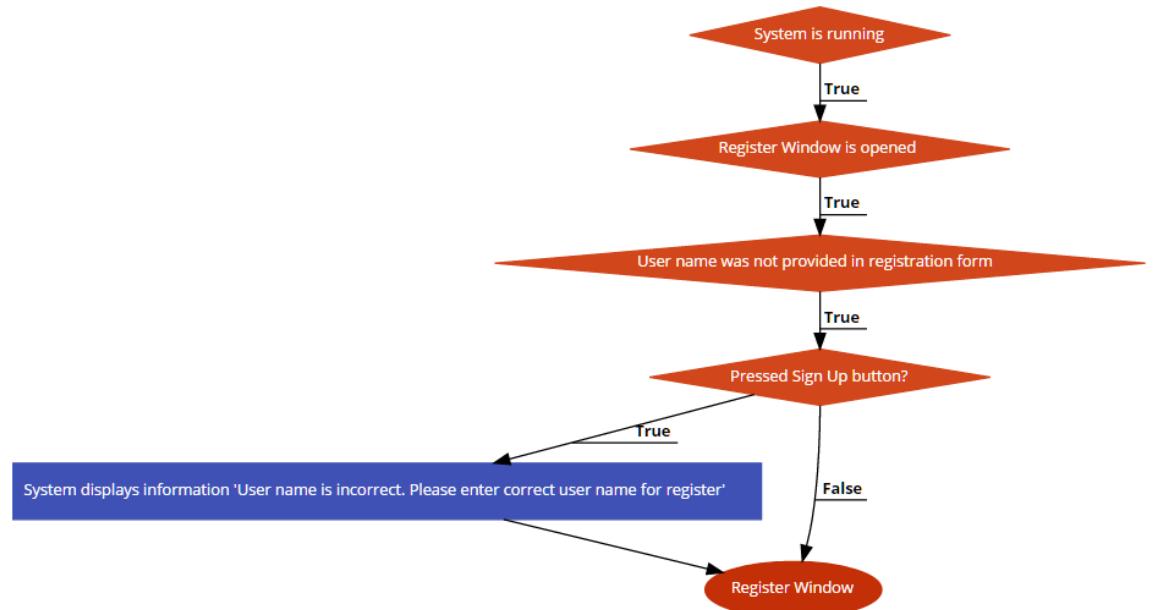


Figure 79: Flow Diagram FR 79

Termination Flow Description

Precondition

System is running, Program runs in Register Window.

Activation

This use case starts when a user pressed Sign Up button after user name was not provided in registration form in Registration Window.

Main flow

- A1. User press Sign Up button
- A2. System displays information "User name is incorrect. Please enter correct user name for register".

Alternate flow

- B1: Sign Up button in Registration Window is not pressed.

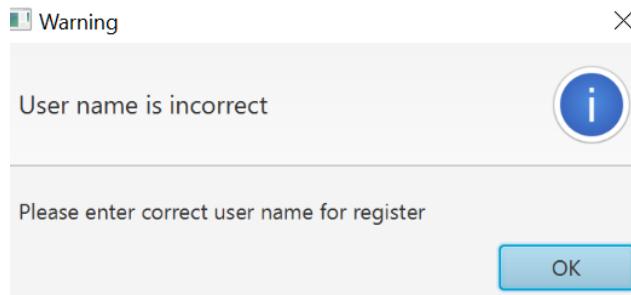
Termination

User not provided user name and pressed Sign Up button.

Post condition

The system displays information ‘User name is incorrect. Please enter correct user name for register’.

Post condition Mock



2.1.2 Data requirements

The application uses the MySQL database system and the SQL language for accessing database. I used MySQL database server version 5.7. In NetBeans IDE was added library 'MySQL JDBC Driver - mysql-connector-java-5.1.39-bin.jar' for connection with MySQL database.

The data requirements are printed in below table

#	Description	Category
DR1	Capability to provide data persistance for the data generated by the app	1
DR2	Capability to perform basic Create, Read, Update, Delete functionality on the data in the various tabels	1
DR3	Capability to provide access to the database for users and admin user	1

Table: Data requirements

The database visualDataJB was created on the basis of the prepared script visualDataJB.sql. To execute the script I used tool MySQL Workbench, version 6.3.

The one of the requirements of the system is to implement a quiz that should help students to test their knowledge of algorithms and data structures. The quiz should be stored in the database. If necessary, the quiz can be expanded with additional questions and answers.

The database visualDataJB consists of 4 tables that allow you to store user data and test questions and answers to a quiz.

The entities and relationship between entities, and the structure of the database visualDataJB is shown in below figures.

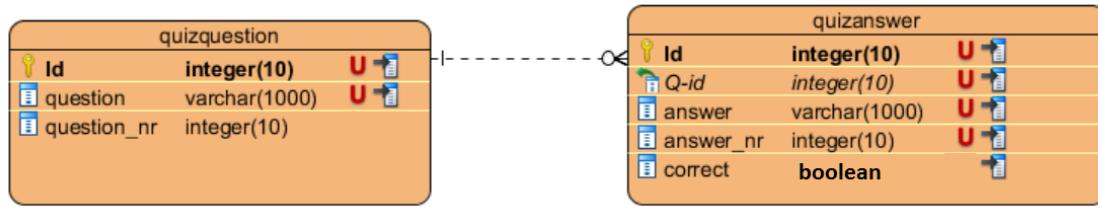


Figure: ER diagram Store Quiz Data

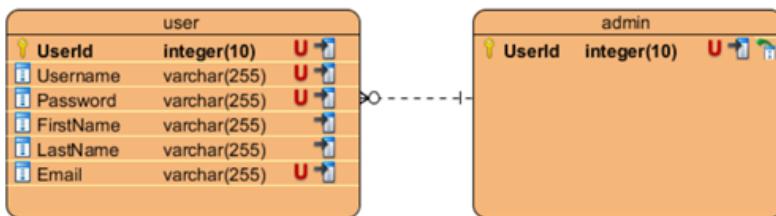


Figure: ER diagram User Data

The database visualDataJB was created in 4 tables: **quizquestion**, **quizanswer**, **user**, **admin**.

The structure of the tables is shown in the figures below.

UserId	Username	Password	FirstName	LastName	Email
1	jacek	jacek	Jacek	Byzdra	jacek@jacek.com
2	guest	guest	NULL	NULL	guest@guest.com
3	test	test	NULL	NULL	test@test.com

Figure: Table user MySQL database visualDataJB

UserId
1

Figure: Table admin MySQL database visualDataJB

Id	Q_Id	answer	answer_nr	correct
1	1	Last in first out	1	1
2	1	First in last out	2	0
3	1	Last in last out	3	0
4	1	First in first out	4	0
5	2	Operations	1	0
6	2	Storage Structures	2	0
7	2	Algorithms	3	0
8	2	None of above	4	1
9	3	linear	1	1
10	3	non linear	2	0
11	3	linked list	3	0
12	3	trees	4	0
13	4	queue	1	0
14	4	stack	2	1
15	4	tree	3	0
16	4	graph	4	0
17	5	finding factorial	1	0
18	5	tower of Hanoi	2	0

Figure: Table quizanswert MySQL database visualDataJB

Id	question	question_nr
1	Stack is also called as	4
2 is not the component of data structure	6
3	A list which displays the relationship of adjacency between elements is...	100
4	The data structure which is one ended is	99
5	Which of the following is an application of stack?	98
6	The time complexity of quick sort is	97
7	In a priority queue, insertion and deletion takes place at ...	96

Figure: Table quizquestion MySQL database visualDataJB

I created the DBConnection.java class, which is responsible for creating a database connection and implements methods to access the database using SQL.

The diagram of the class DBConnection.java is presented in below figure.

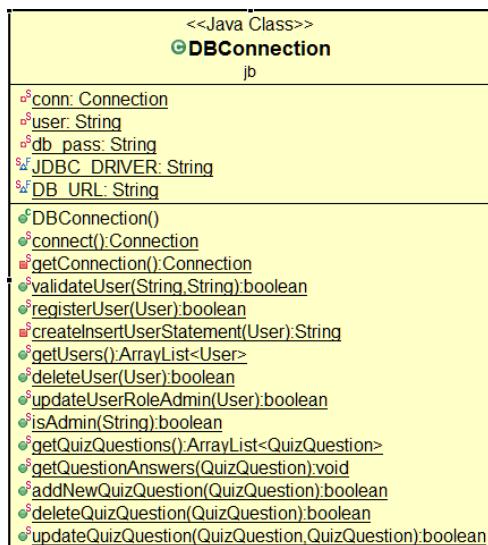


Figure: DBConnection.java class diagram

The methods of this class perform read, write and delete the data of the program users, and read, write and delete the data of the questions and answers from the

database visualDataJB . The methods are static that do not require creation of instance of the class DBConnection.

The method used in the DBConnection.java class are listed below.

DBConnection.Connect() – create database connection,

validateUser() – checks if username and password exist in database,

registerUser() – insert new user to database,

getUsers() – return all users from database,

deleteUser() – delete single user from database,

updateUserRoleAdmin() – set or delete user id in the table Admin. This method updates role of Admin,

isAdmin() – checks if user has role Admin,

getQuizQuestions() – return all question from database,

getQuestionAnswers() – return all answers for a single question,

addNewQuizQuestion() – add new questions with its answers to the database,

deleteQuizQuestion() – delete single question with all its answers from the database,

updateQuizQuestion() – save changes in the single question or its answers,

The table user keeps user data. In the application class user is responsible for user data, and keeps structure for singel user with attribute: Username, Password, Email, UserId.

The class QuizQuestion.java corresponds with table question and keeps the structure for single quiz question. The quiz corresponds with the class QuizQuestion.java where answers are kept in the List of Strings.

One table ‘Questions’ holds questions to the quiz. The Table ‘answers’ stores possible answers.

MySQL 5.7 database visualDataJB keeps records of the registered users and the records of all quiz questions and answers. The quiz can be expanded with additional questions and answers provided to the database.

2.1.3 User requirements

The user requirements presents requirements of the system from user perspective. It includes services, knowledge required to operate the application. The application presents visualisation and animation of algorithms and data structures, designed and coded in Java language.

The level of presented information requires from user basic knowledge of data structures and algorithms and basic knowledge of Java language. To read instructions and notes presented in the application user should handle English language on medium level.

2.1.4 Environmental requirements

The environment requirements describe the equipment and software required for the development and testing of the app.

The list of this environmental requirements is presented below. All requirements are category 1.

1. ER1 – Computer PC with installed Windows Client, or Linux, or OS X Operation System which comply requirements :

Windows Client				
Platform	CPU Architecture	Version	Introduced In	Notes
Windows 10	x86 (32-bit)		1.8.0_51	
Windows 10	x64 (64-bit)		1.8.0_51	
Windows 8.x	x86 (32-bit)		1.8.0	
Windows 8.x	x64 (64-bit)		1.8.0	
Windows 7	x86 (32-bit)	SP1	1.8.0	
Windows 7	x64 (64-bit)	SP1	1.8.0	
Linux				
Platform	CPU Architecture	Version	Introduced In	Notes

Linux	x64 (64-bit)	7.x	1.8.0_20	
OS X				
Platform	CPU Architecture	Version	Introduced In	Notes
OS X	x64	10.9 and above	1.8.0	Only 64-bit JRE is supported
OS X	x64	10.8.3+	1.8.0	Only 64-bit JRE is supported

2. ER2 –Installed Java JDK 1.8 or higher on-top on Operation System.
3. ER3 - Installed MySQL 5.7 or higher on-top on Operation System.
4. ER4 - PC computer which comply requirements

JavaFX Graphics Support	
<p>For JavaFX applications to take advantage of the new hardware acceleration pipeline provided by JavaFX, system must feature one of the graphics cards below. Otherwise JavaFX will default to the Java2D software pipeline. the Java2D software pipeline</p>	
Graphics card	Supported Graphics Processing Units (GPUs)
NVIDIA	<p>Mobile GPUs: GeForce 8M and 100M series or higher, NVS 2100M series or higher, and Mobility Quadro FX 300M series or higher</p> <p>Desktop GPUs: GeForce 8 and 100 series or higher</p> <p>Workstation GPUs: Quadro FX 300 series or higher</p>
ATI	<p>Mobile GPUs: Mobility Radeon HD 3000, 4000, and 5000 series</p> <p>Desktop GPUs: Radeon HD 2400, 3000, 4000, 5000, and 6000 series</p>
Intel	<p>Mobile GPUs: GMA 4500MHD and GMA HD</p> <p>Desktop GPUs: GMA 4500 and GMA HD</p>
<p>JavaFX supports graphic hardware acceleration on any certified OS X system. On Linux platforms, graphic hardware acceleration is only supported for Nvidia cards (proprietary drivers only).</p>	

2.1.5 Usability requirements

The software application was designed as learning tool for variety of students which learn algorithms and data structures. It was implemented with attractive

features and attractive and interactive Graphical User Interface. The list of usability requirements is listed below:

No	Description of Usability requirement	Category
UBR1	Appeal and value to user groups	1
UBR2	Effective features of the system	1
UBR3	Efficiency of the system	1
UBR4	Interactive, rich, and responsive GUI	1

2.2 Design and Architecture

The system was designed in Java and JavaFX language to provide an interactive environment for learning. The program allows the user to display on the monitor screen data structures: graphs and trees, sorting algorithms, and interactive quiz, and hints about the data structures and algorithms. The user may perform basic operations on the graphs and trees, change the size of graph, tree, and size of the list generated for searching algorithms. The program visualize and animate the searching algorithms, and the algorithms based on the tree and the graph data structure.

The software architecture of the program was based on the Model View Controller (MVC) pattern.

The view was designed on FXML markup script language. The controller was focused on logic, and model was designed as main model in the system. JavaFX Cascading Style Sheet CSS was used to apply styles to view.

The structure of the files in the system is presented in below diagram.

Visual Data

Models	Views	Controllers	Database
AlgorithmDescription.java BST.java BSTNode.java DBConnection.java Graph.java GraphEdge.java GraphNode.java LoginUser.java QuizQuestion.java UsedNumberList.java WelcomeApp.java	TreeBSTView.fxml admin.fxml description.fxml graph.fxml home.fxml login.fxml menu.fxml quiz.fxml register.fxml sort.fxml welcome.fxml style.css	AdminController.java DescriptionController.java GraphController.java HomeController.java LoginController.java MenuController.java QuizController.java RegisterController.java SortController.java Sort.java TreeBSTViewController.java WelcomeController.java	MySQL database MySQL JDBC driver

Figure: Structure of Visual Data system

The graphic user interface of the project was built entirely with JavaFX running on Java 8. The fxml JavaFX markup language files used for views, comprise hierarchical structure of all GUI elements. The fxml files are user interfaces of JavaFX application. JavaFX Cascading Style Sheet (CSS) was used to provide definitions for style used in the project. The style is used by controls and objects in the scene to change color and fonts. The main application class was created in the file WelcomeApp.java.

The program consists of several MVC sections:

- Section Welcome,
- Section Login,
- Section Register,

- Section Menu,
- Section Admin,
- Section Quiz,
- Section Sorts Bars,
- Section Sorts Panels,
- Section Binary Search Tree,
- Section Graph.

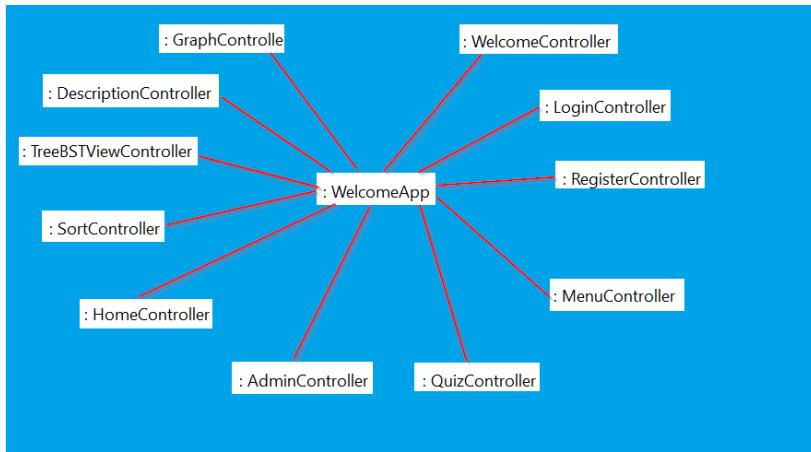


Figure: MVC sections Visual Data

2.2.1 Design and Architecture – Welcome Section

The section Welcome based on the files: WelcomeApp.java, WelcomeController.java, welcome.fxml is initiated immediately after startup of the program.

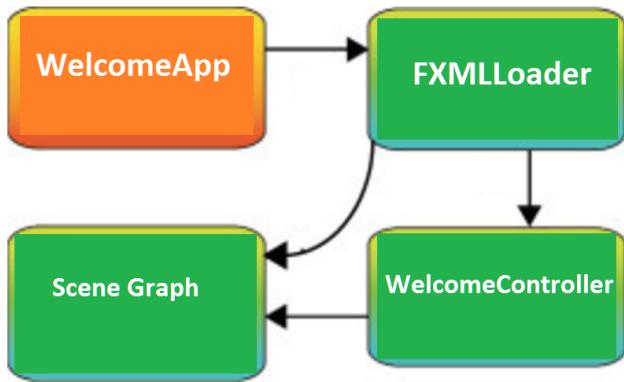


Figure: Structure of JavaFX WelcomeApp Application – Welcome Controller

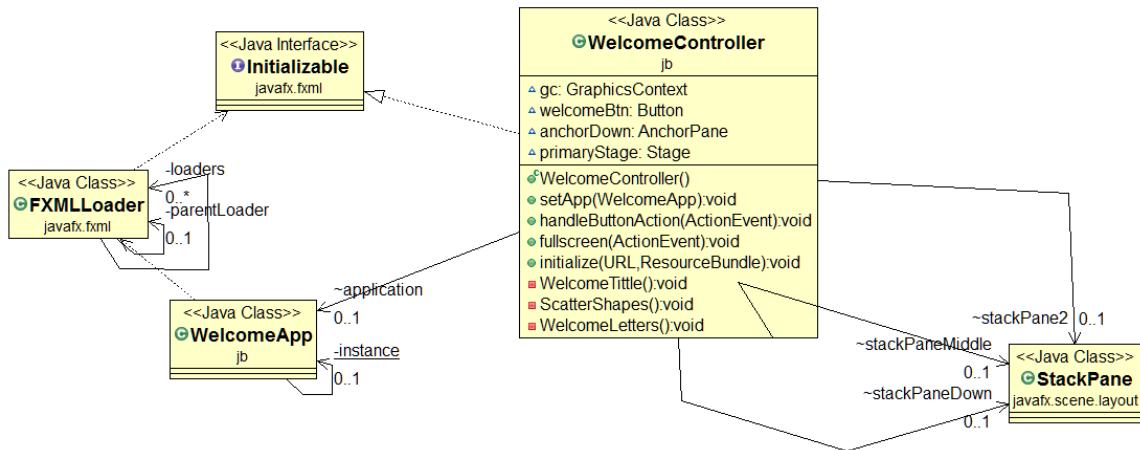


Figure: Welcome Diagram Class

The design of the graph objects and the GUI interface of the program was based on JavaFX FXML markup language. The views were based on separate FXML files.

The structure of FXML used in the project includes a class instance and a properties of class instance: (Introduction to FXML, Release: JavaFX 8.0, www.oracle.com).

The FXMLLoader class loads an FXML source file, creates instance of a class, when the tag of fxml file begins with uppercase, and returns the resulting graph elements. A property of a class instance in FXML file is an attribute used to configure the properties of the class.

The structure of welcomejb.fxml file used in project is presented in below figure.

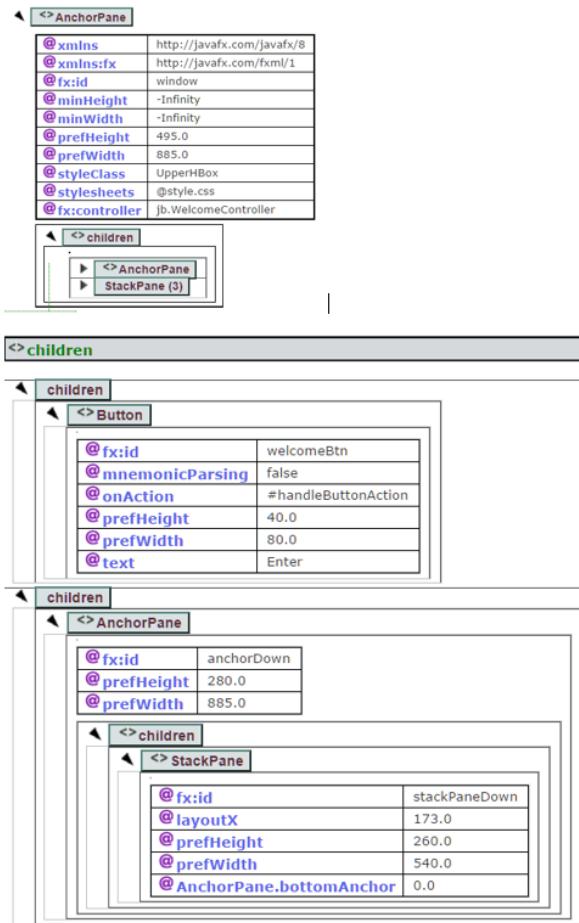


Figure: welcomejb.fxml file structure

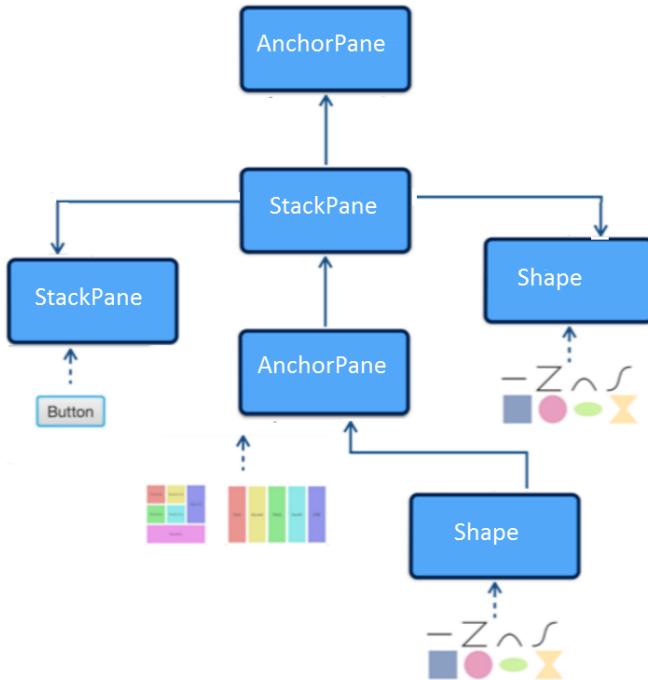


Figure: welcomejb.fxml node class hierarchy

All UI components extend the base class Node.

The AnchorPane, StackPane node controls are used to show the scene of the graph objects and the shape nodes to show animation effects. The Button "welcomeBtn" node control is used to invoke the switchover to login fxml view.

The instance of class AnchorPane is imported from javafx.scene.layout.Pane package. The styles are defined in the style.css file. The controller WelcomeController is associated with welcomejb.fxml document. The children property represents the child nodes in AnchorPane class. Each sub-element of the children element is added to the list returned by AnchorPane. The fxml markup creates a Button id="welcomeBtn" which after click invokes and starts method handleButtonAction() from the controller WelcomeController. The method handleButtonAction invokes gotoLogin() method from WelcomeApp.java main class.

```

@FXML
public void handleButtonAction(ActionEvent event) throws IOException{
    if(event.getSource()==welcomeBtn){
        WelcomeApp.getInstance().gotoLogin();
    }
}

```

The method `gotoLogin()` replaces current `welcomejb.fxml` Scene content to `login.fxml` Scene content.

```

public void gotoLogin(){
    try {
        replaceSceneContent("login.fxml");
        stage.setTitle("Visual data - Login");
    } catch (Exception ex) {
        Logger.getLogger(WelcomeApp.class.getName()).log(Level.SEVERE,
null, ex);
    }
}

```

In the Welcome section were applied 3 animation methods for greeting user.

The methods draw and add graphic objects to StackPane of `welcome.fxml` view.

In the following method, the color of the title changes from blue to violet and back again in a continuous cycle.

```

private void WelcomeTittle()
{
    stackPane2.getChildren().clear();

    final Text txt = new Text(10, 50, "Welcome In Visual Data");
    txt.setFont(Font.font ("Verdana", 48));
    txt.setFill(Color.RED);
    stackPane2.getChildren().add(txt);

    StrokeTransition st = new StrokeTransition();
    st.setDuration(Duration.seconds(20));
    //st.setDelay(Duration.seconds(.5));
    st.setShape(txt);
    st.setFromValue(Color.BLUE);
    st.setToValue(Color.VIOLET);
    st.setCycleCount(Timeline.INDEFINITE);
}

```

```

        st.setAutoReverse(true);
        st.setRate(10);
        st.play();
    }
}

```

The method ScatterShapes is animation method which generates random circles nodes with different size and colors, and position the circle nodes randomly in the scene. The circle are placed in AnchorPane control node and displayed in the screen.

```

private void ScatterShapes(){
    Random random = new Random(System.currentTimeMillis());
    for (int i = 0; i < 500; i++) {
        Circle circle = new Circle(random.nextInt(1920),
random.nextInt(250),random.nextInt(25));
        circle.setFill(Color.rgb(random.nextInt(255),random.nextInt(255),
random.nextInt(255), 0.5));
        anchorDown.getChildren().add(circle);
    }
}

```

The WelcomeLetters method animate movement of text Node elements in the screen. The text Nodes are added to the paths. The position of the Nodes is changed in MoveTo() and LineTo() methods. The sequence of changing Node's position is animated according to pathTransition() method.

```

private void WelcomeLetters()
{
    stackPaneDown.getChildren().clear();

    final Text txt1 = new Text(10, 50, "Welcome");
    final Text txt3 = new Text(10, 50, "In ");
    final Text txt5 = new Text(10, 50, "Visual ");
    final Text txt7 = new Text(10, 50, "Data ");

    txt1.setFont(Font.font ("Rockwell", 48));
    txt3.setFont(Font.font ("Verdana", 48));
    txt5.setFont(Font.font ("Rockwell", 48));
    txt7.setFont(Font.font ("Verdana", 48));

    txt1.setFill(Color.RED);
    txt3.setFill(Color.BLUEVIOLET);
}

```

```

txt5.setFill(Color.CORAL);
txt7.setFill(Color.GOLD);

final Path path = new Path();
path.getElements().add(new MoveTo(200, 0));
path.getElements().add(new LineTo(880, 0));

path.setOpacity(0.0);

stackPaneDown.getChildren().add(path);
stackPaneDown.getChildren().add(txt1);
final PathTransition pathTransition = new PathTransition();

pathTransition.setDuration(Duration.seconds(8.0));
pathTransition.setDelay(Duration.seconds(.5));
pathTransition.setRate(0.3);
pathTransition.setPath(path);
pathTransition.setNode(txt1);

pathTransition.setOrientation(PathTransition.OrientationType.ORTHOGONAL_T
O_TANGENT);
pathTransition.setCycleCount(Timeline.INDEFINITE);
pathTransition.setAutoReverse(true);
pathTransition.play();

```

The below figures present welcome view when the WelcomeApp is started.

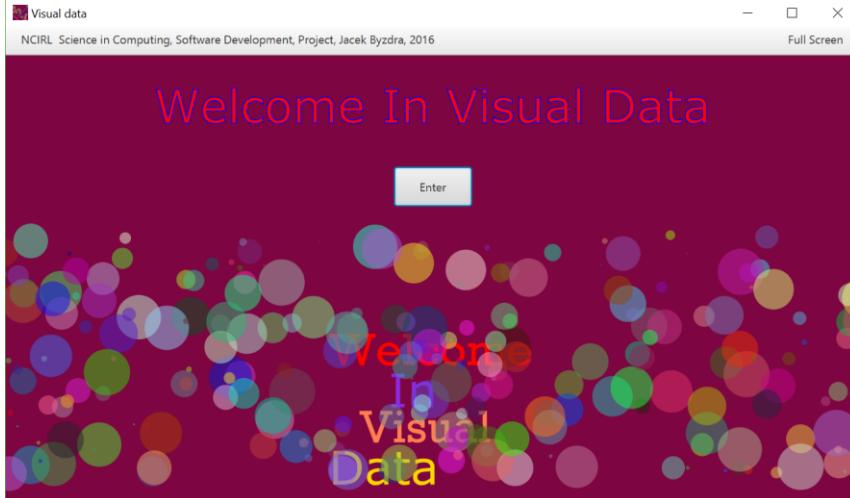


Figure: Welcome view 1

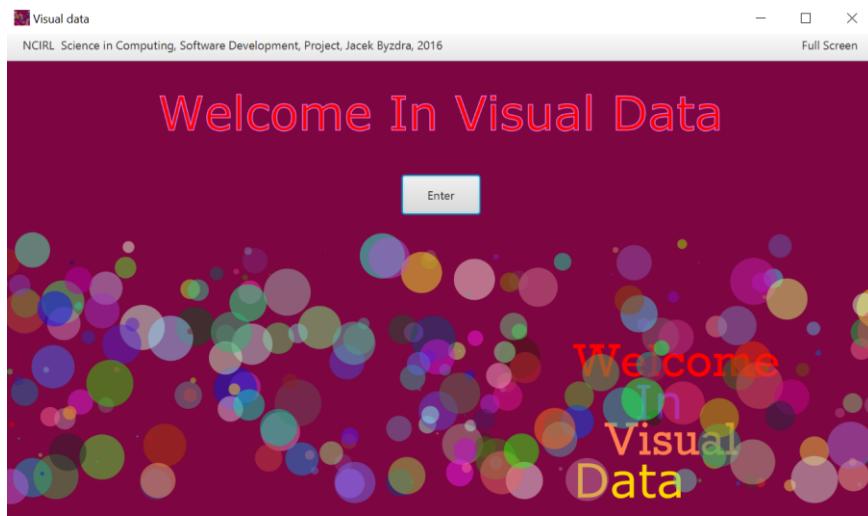


Figure: Welcome view 2

2.2.2 Design and Architecture – Login Section

The method `handleButtonAction` in Welcome view invokes `gotoLogin()` method from `WelcomeApp.java` main class.

The MVC structure of the login section is composed of file:

`LoginUser.java`, `WelcomeApp.java`, `LoginController.java`, `login.fxml`.

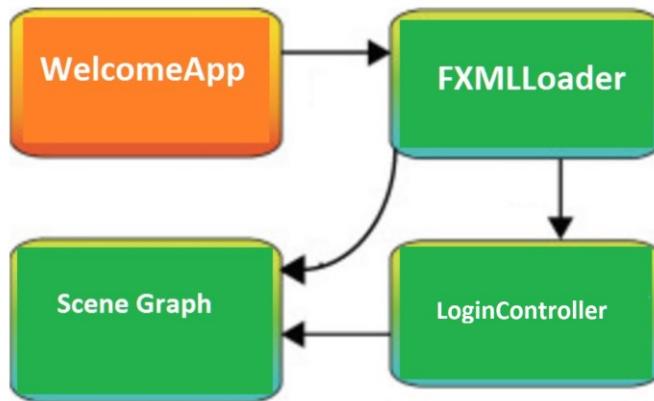


Figure: Structure of JavaFX WelcomeApp Application – Login Controller

In the login section MySQL database is used to validate user name and password.

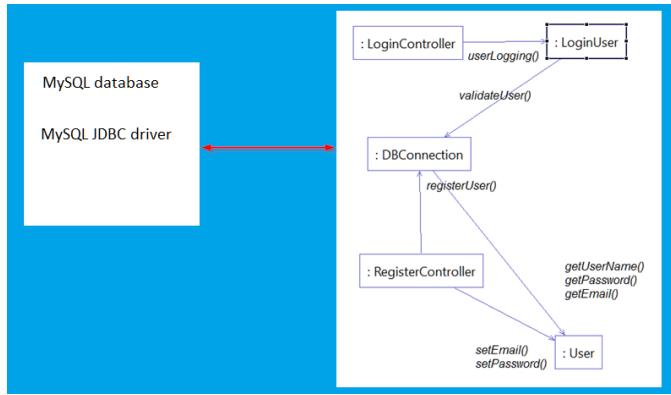


Figure: Login section Visual Data

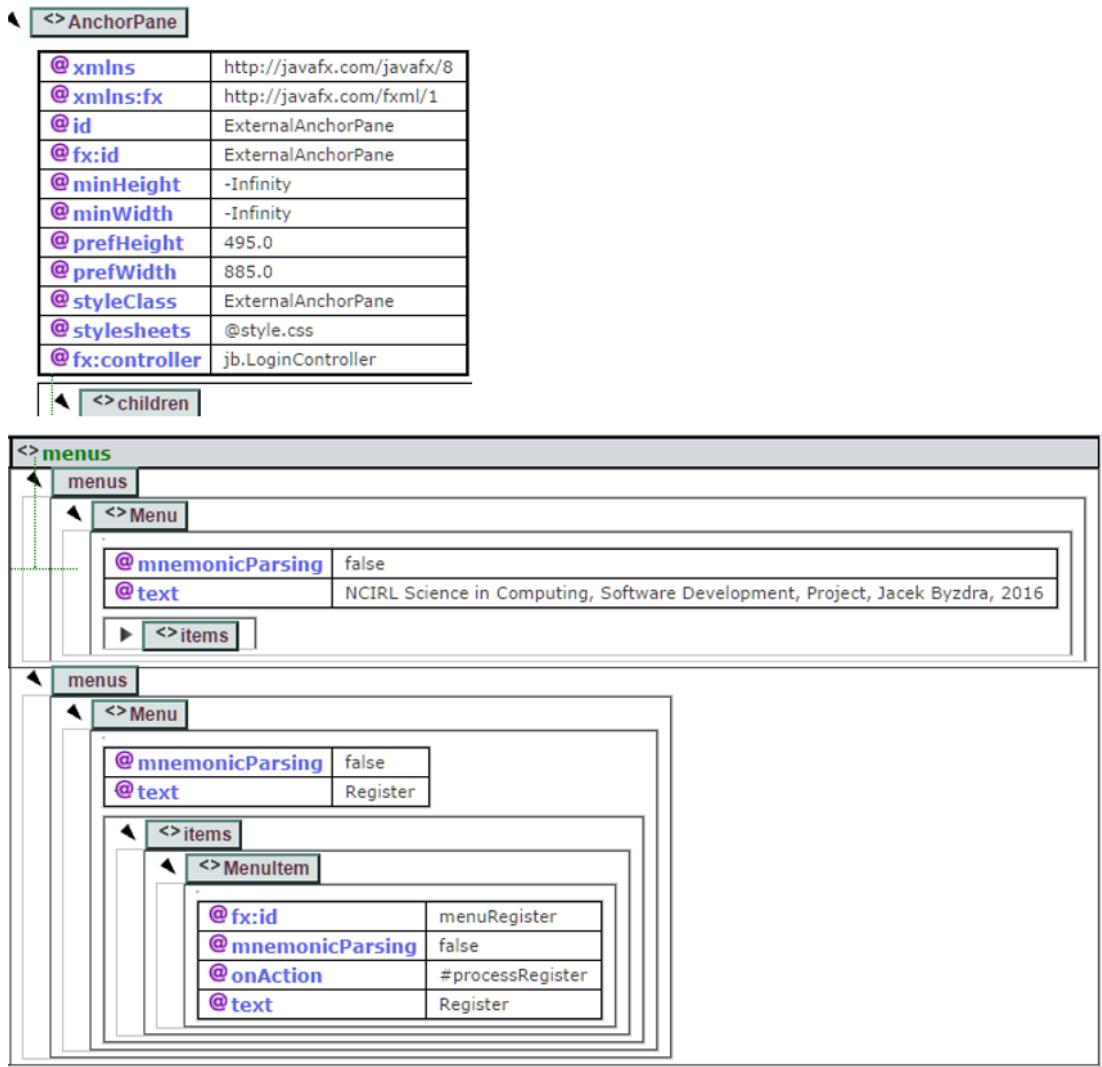


Figure: login.fxml structure

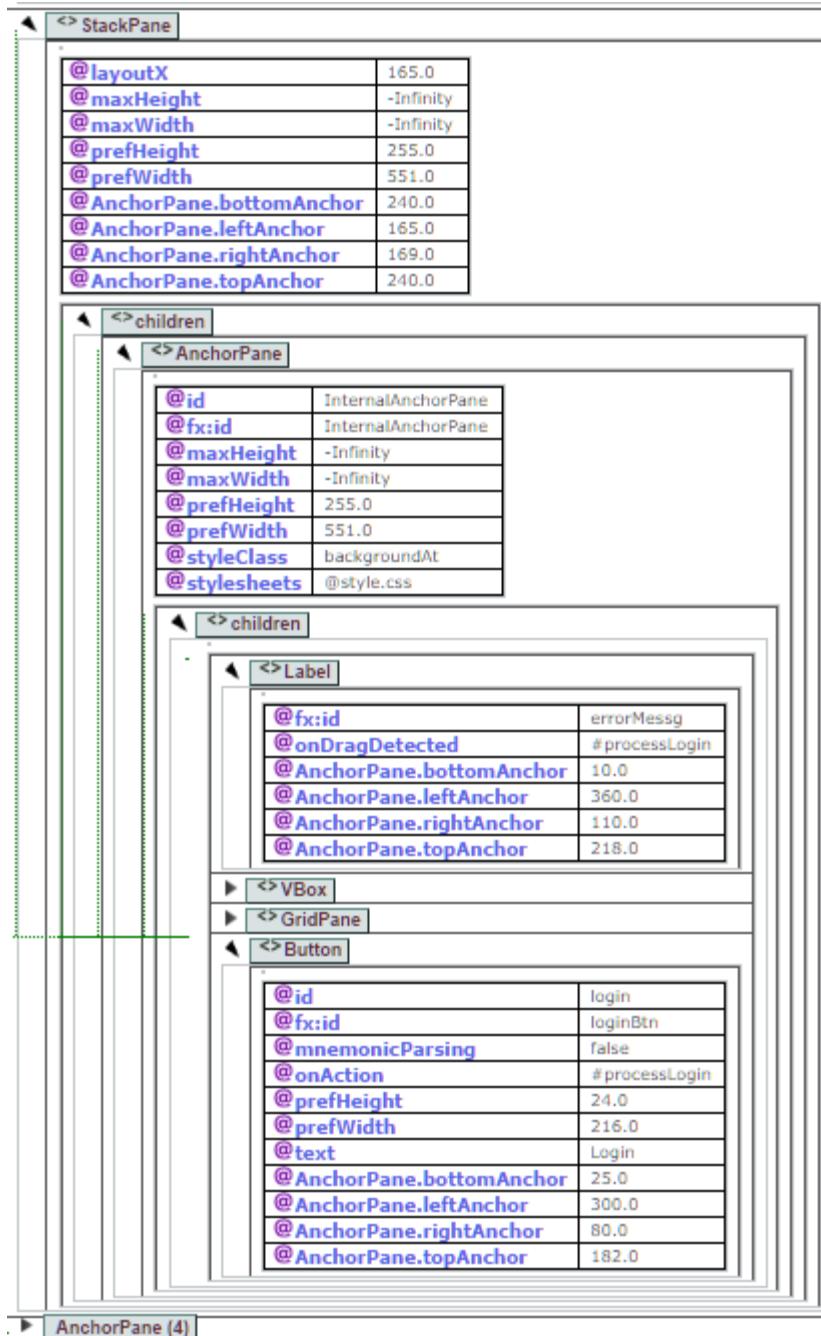


Figure: login.fxml structure cont.

The instance of class `AnchorPane` is imported from `javafx.scene.layout.Pane` package. The styles are defined in `style.css` file. The controller `LoginController` is associated with `login.fxml` document. The `children` property represents the child nodes in `AnchorPane` class. Each sub-element of the `children` element is added

to the list returned by AnchorPane. The fxml markup creates a Button id="loginBtn" , and Label 'Password' , and Label 'UserName'.

When a password and username are filled, and user pressed login button the method userLogging () is invoked from LoginUser.java class.

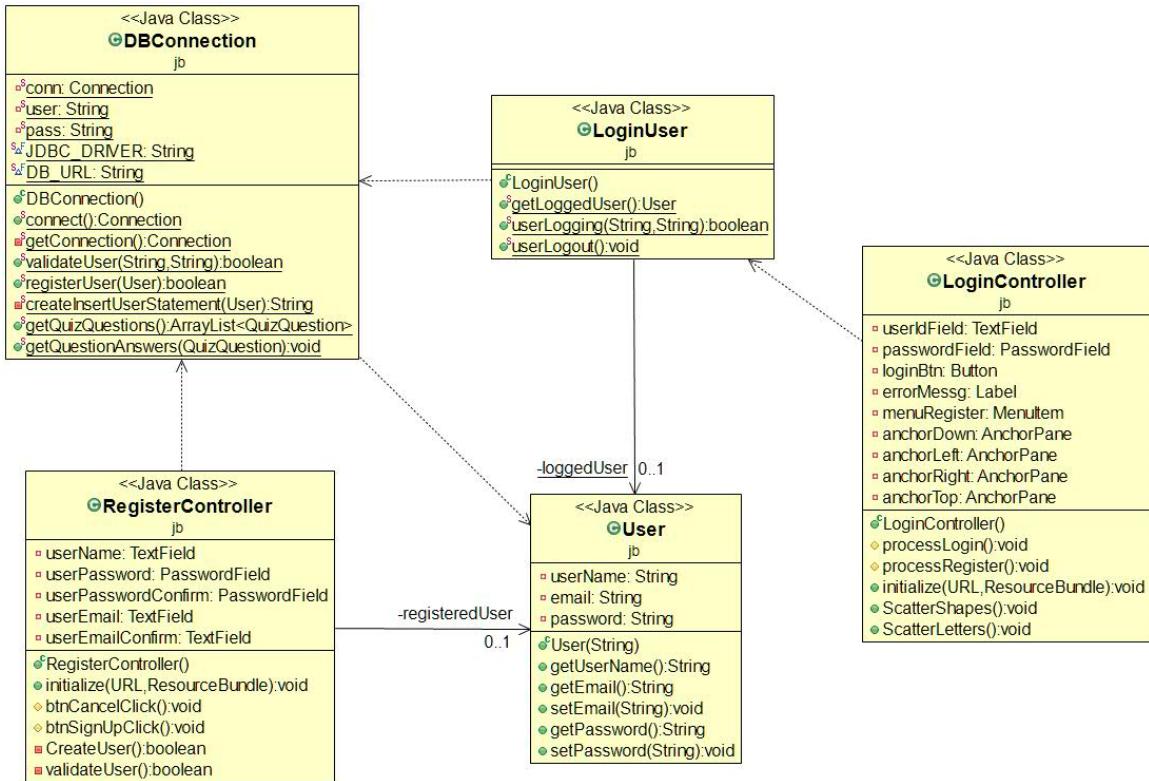


Figure: Login section class

In the method userLogging DBConnection validate userName and password in MySQL database, and after successful validation the system invokes method gotoMenu() from main WelcomeApp.java class.

```

public static boolean userLogging(String userId, String password){
    if (DBConnection.validateUser(userId, password)) {
        loggedUser = new User(userId);
        loggedUser.setAdmin(DBConnection.isAdmin(userId));
        WelcomeApp.getInstance().gotoMenu();
        return true;
    } else {

```

```

        return false;
    }
}

```

In the login section user may invoke register methods to register user data in MySQL database. In top right position of the login section user may choose MenuItem ‘menuRegister’ which invokes processRegister() method from LoginController.java class.

```

@FXML protected void processRegister(){
    WelcomeApp.getInstance().gotoRegister();
}

```

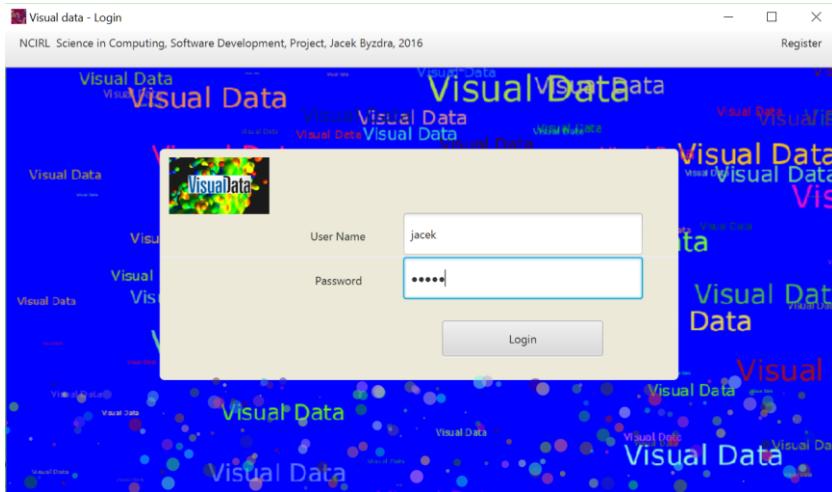


Figure: Login view

2.2.3 Design and Architecture – Register Section

The MVC structure of the register section is composed of file:

WelcomeApp.java, RegisterController.java, register.fxml.

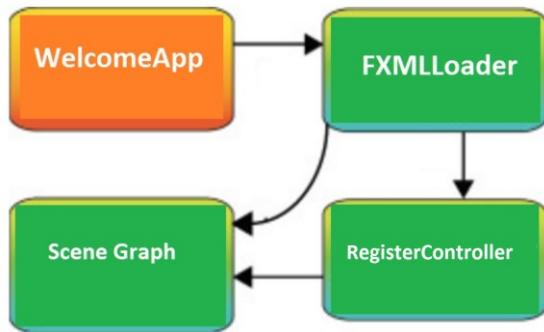


Figure: Structure of JavaFX WelcomeApp Application – Register Controller

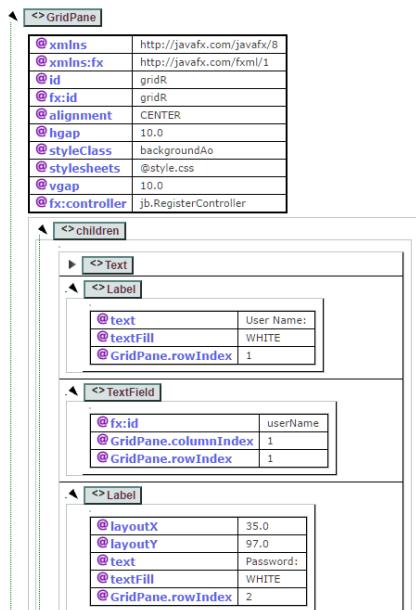


Figure: register.fxml structure

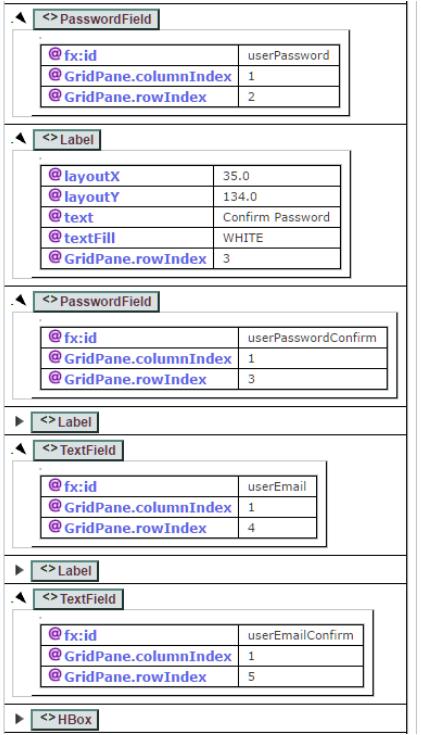


Figure: register.fxml structure cont.

The instance of class `GridPane` is imported from `javafx.scene.layout.Pane` package. The styles are defined in `style.css` file. The controller `RegisterController` is associated with `register.fxml` document. The `children` property represents the child nodes in `GridPane` class. Each sub-element of the `children` element is added to the list returned by `GridPane`. The fxml markup creates several Labels for user: ‘User Name’ , ‘Password’, ‘Confirm Password’, ‘email’, ‘Confirm email’. When all fields are filled and register button is pressed the system invokes method `CreateUser()` in `RegisterController.java` class.

```
@FXML protected void btnSignUpClick(){
    if(CreateUser()){
        WelcomeApp.getInstance().closeRegister();
    }
}

private boolean CreateUser(){
    if(!validateUser()){
        return false;
    }
}
```

```

}

registeredUser = new User(userName.getText().trim());
registeredUser.setEmail(userEmail.getText());
registeredUser.setPassword(userPassword.getText());

// send to database:
if (DBConnection.registerUser(registeredUser)) {
    // if success:
    WelcomeApp.getInstance().showMessage("Congratulation!", "You are
registered in this application", "Please log in now");
    return true;
}
// if unsuccess:
WelcomeApp.getInstance().showMessage("Warning", "Registration error",
"Please try again");
return false;
}

```

In the method CreateUser() new user is created in MySQL database. The new user record is associated with userName, userPassword and userEmail. After registration user may login to the system.

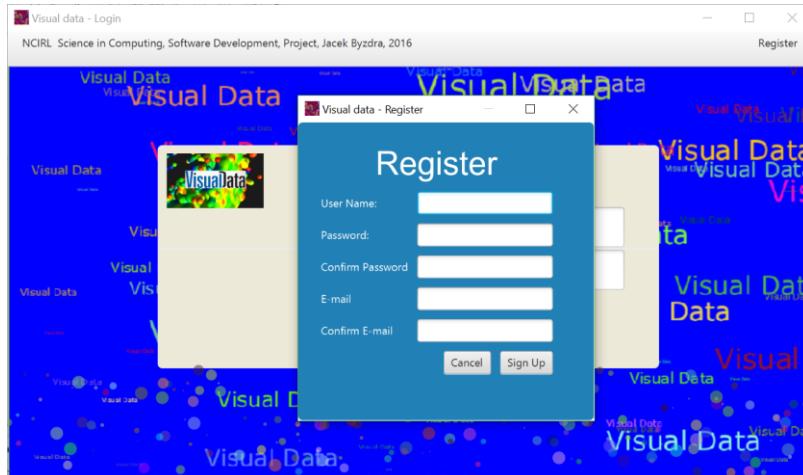


Figure: Register view

2.2.4 Design and Architecture – Menu Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

QuizQuestion.java

MenuController.java

QuizController.java

DescriptionController.java

admin.fxml

description.fxml

menu.fxml

quiz.fxml

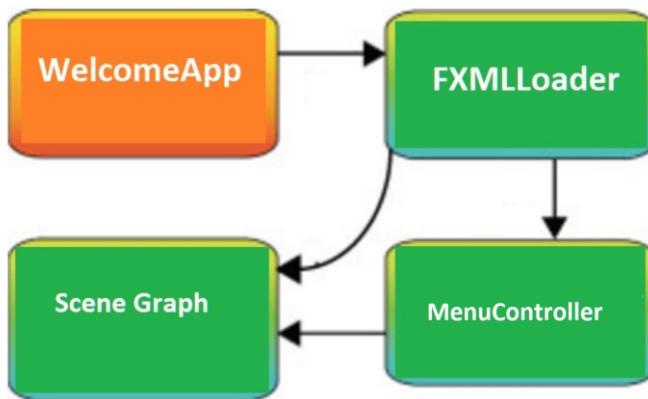


Figure: Structure of JavaFX WelcomeApp Application – Menu Controller

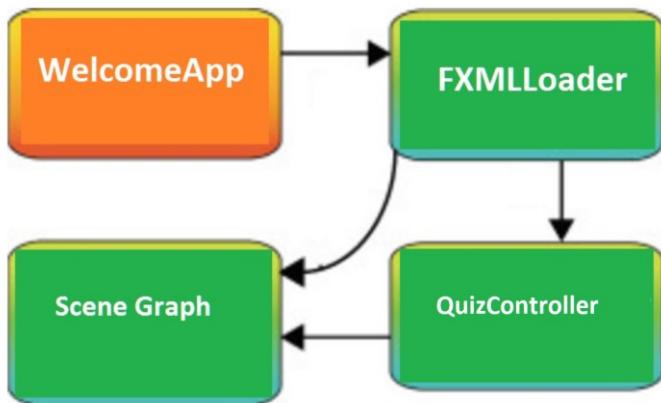


Figure: Structure of JavaFX WelcomeApp Application – QuizController

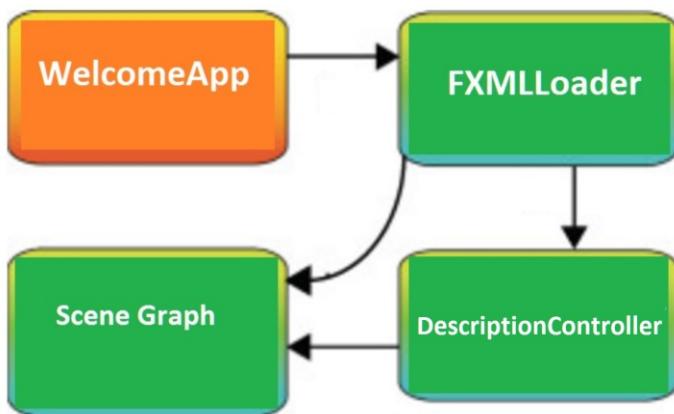


Figure: Structure of JavaFX WelcomeApp Application – Description Controller

The structure of menu.fxml file is shown in figure below:

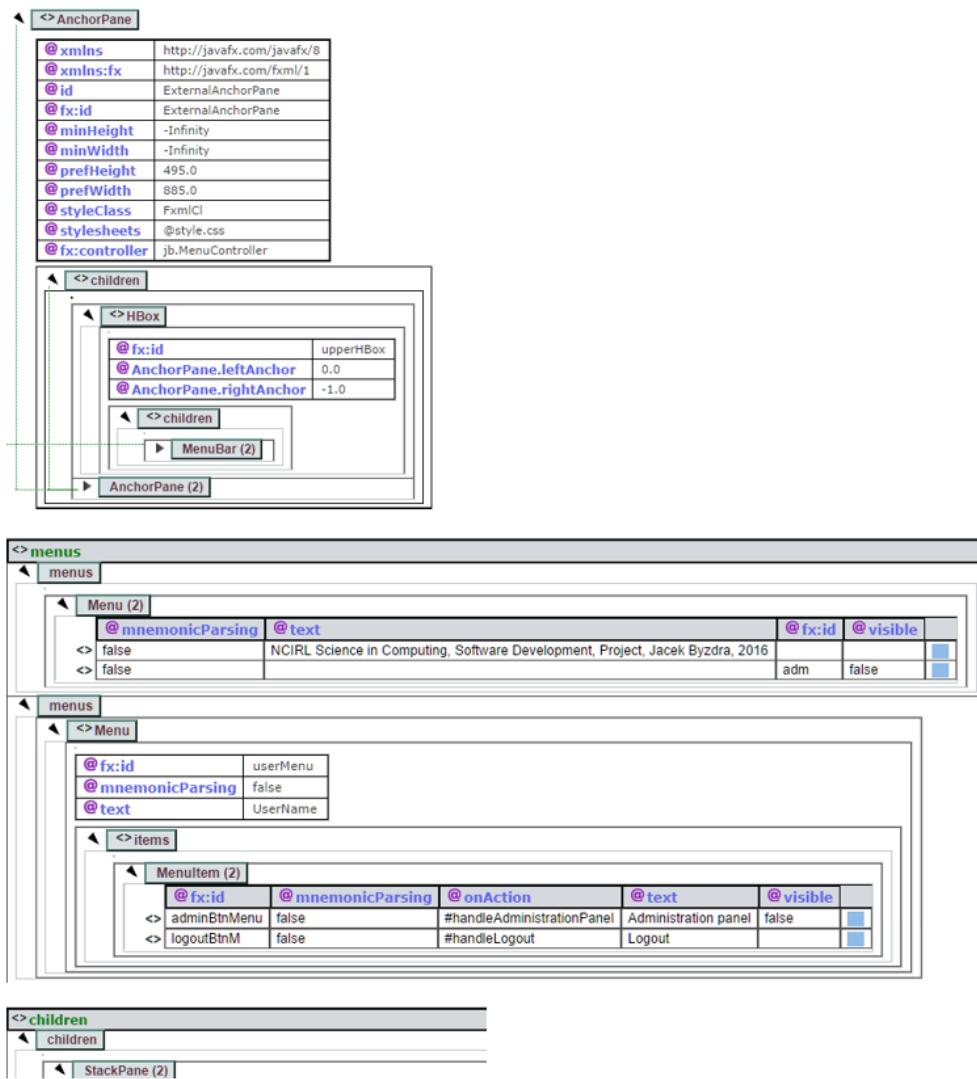


Figure: menu.fxml structure

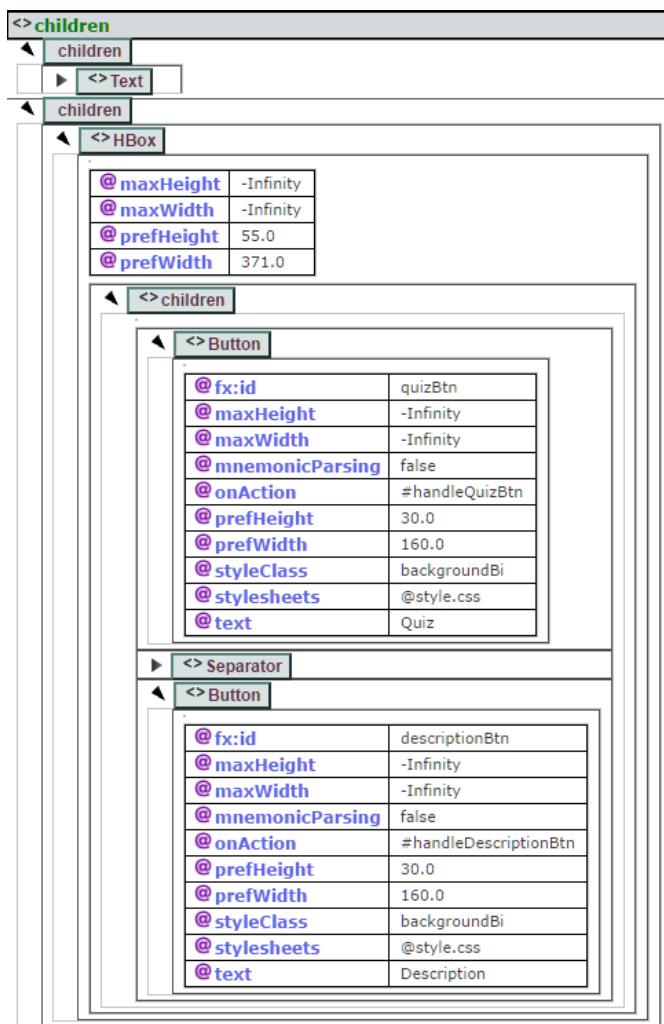


Figure: menu.fxml structure cont.

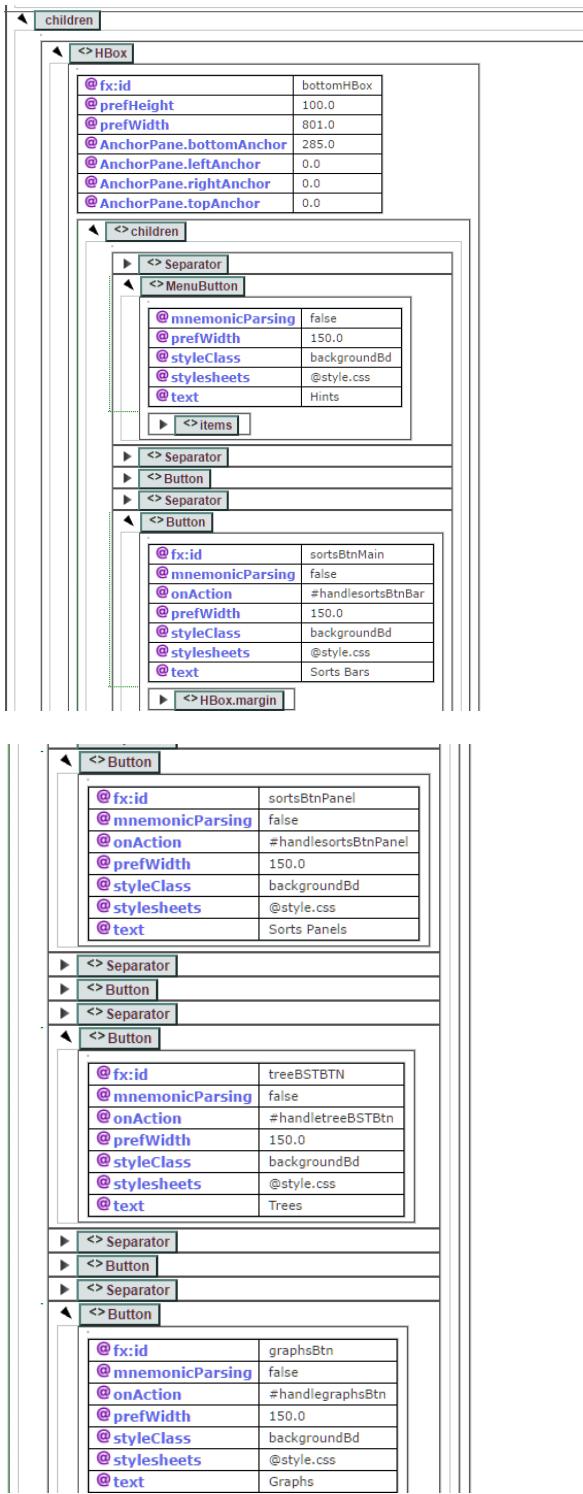


Figure: menu.fxml structure cont.

The instance of the class AnchorPane in menu.fxml is imported from javafx.scene.layout.Pane package. The styles are defined in style.css file. The controller of the class is MenuController.java in jb package. The children property represents the child nodes in AnchorPane class. The menu.fxml creates severals Buttons:id=" sortsBtnMain", id="sortsBtnPanel", id="quizBtn", id="descriptionBtn", id=" treeBSTBTN", id="graphsBtn", id="logoutBtnM", id=" adminBtnMenu".

The Button 'sortsBTNMain' when click invokes handlesortsBtnBar() method from MenuController.

```
@FXML  
public void handlesortsBtnBar(ActionEvent event){  
    if(event.getSource()==sortsBtnMain){  
        WelcomeApp.getInstance().gotoHome();  
    }  
}
```

The method handlesortsBtnBar() when is initiated invokes method gotoHome() from WelcomeApp.java main class.

```
public void gotoHome() {  
    try {  
        replaceSceneContent("home.fxml");  
        stage.setTitle("Visual data - Sort (Bars)");  
    } catch (Exception ex) {  
        Logger.getLogger(WelcomeApp.class.getName()).log(Level.SEVERE,  
null, ex);  
    }  
}
```

The method gotoHome() replaces current menu.fxml Scene content to home.fxml Scene content, which is sorts bars main view .

In similar way are designed the Buttons: sortsBtnPanel, treeBSTBTN and graphsBtn.

The button sortsBtnPanel when click invokes replacement of current menu.fxml Scene content to sort.fxml Scene , which is sorts panels main view.

```
public void gotoSort() {  
    try {  
        replaceSceneContent("sort.fxml");  
        stage.setTitle("Visual data - Sort (Panels)");  
    } catch (Exception ex) {
```

```
        Logger.getLogger(WelcomeApp.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
```

The button treeBSTBTN when click invokes replacement of current menu.fxml Scene content to TreeBSTView.fxml Scene, which is Binary Search Tree main view.

```
public void gotoTreeBST() {
    try {
        replaceSceneContent("TreeBSTView.fxml");
        stage.setTitle("Visual data - Trees");
    } catch (Exception ex) {
        Logger.getLogger(WelcomeApp.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
```

The button graphsBtn when click invokes replacement of current menu.fxml Scene content to graph.fxml Scene, which is Graph main view.

```
public void gotoGraph() {
    try {
        replaceSceneContent("graph.fxml");
        stage.setTitle("Visual data - Graphs");
    } catch (Exception ex) {
        Logger.getLogger(WelcomeApp.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
```

The Menu view is presented in below figure.

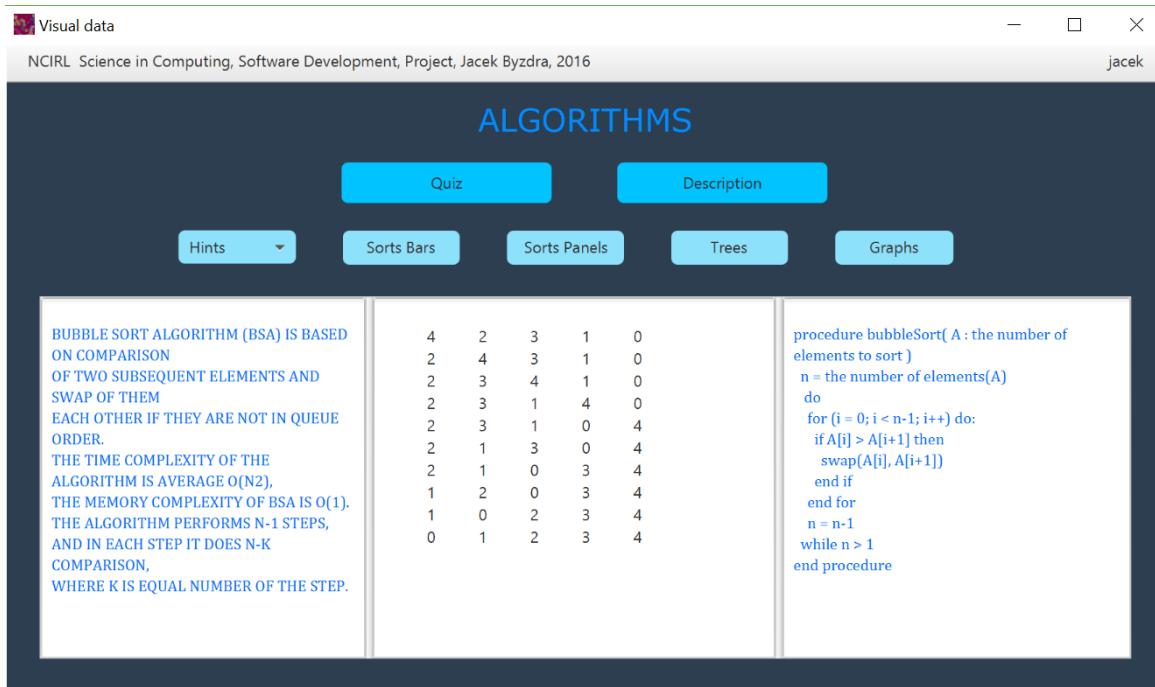


Figure: menu.fxml view

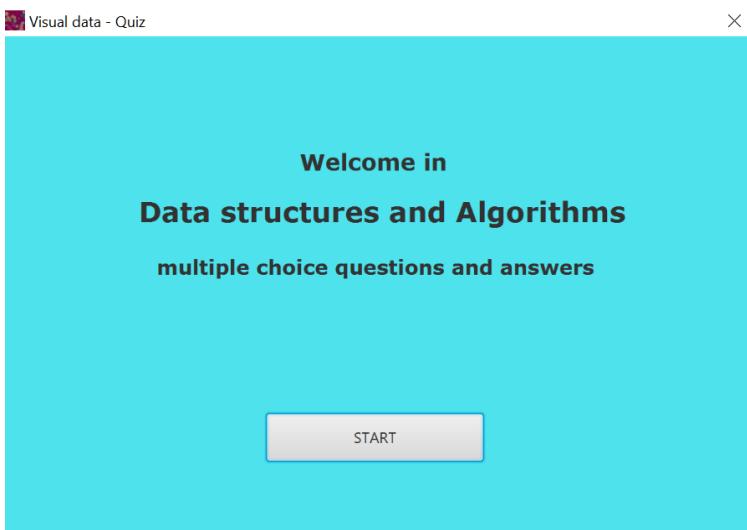


Figure: Quiz view

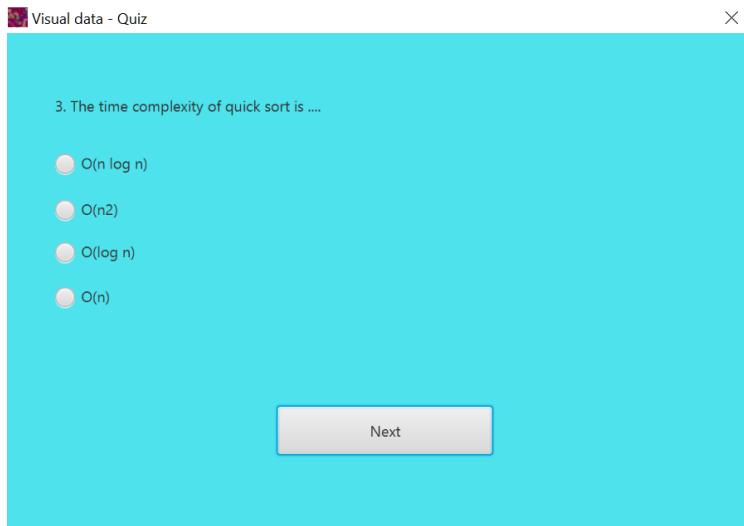


Figure: Quiz view cont

No.	Question
1 is not the component of data structure
2	A list which displays the relationship of adjacency between elements is said to be
3	In a priority queue, insertion and deletion takes place at ...
4	Stack is also called as
5	The data structure which is one ended is

Quiz question:

Answer A: _____

Answer B: _____

Answer C: _____

Answer D: _____

Figure: Quiz Administration view

2.2.5 Design and Architecture – Bars Sorts Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

HomeController.java

home.fxml

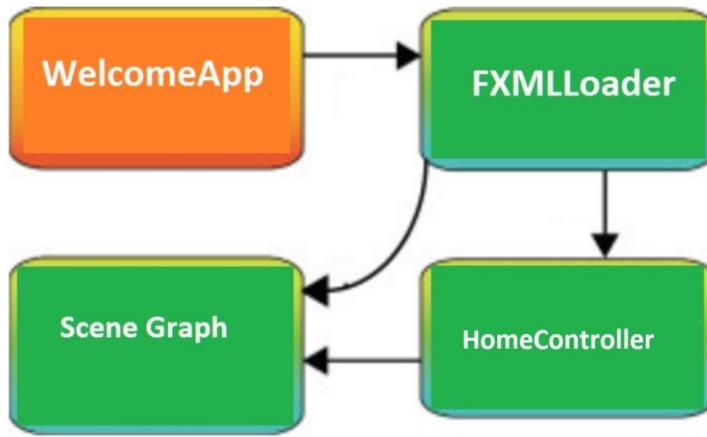


Figure: Structure of JavaFX WelcomeApp Application – Home Controller

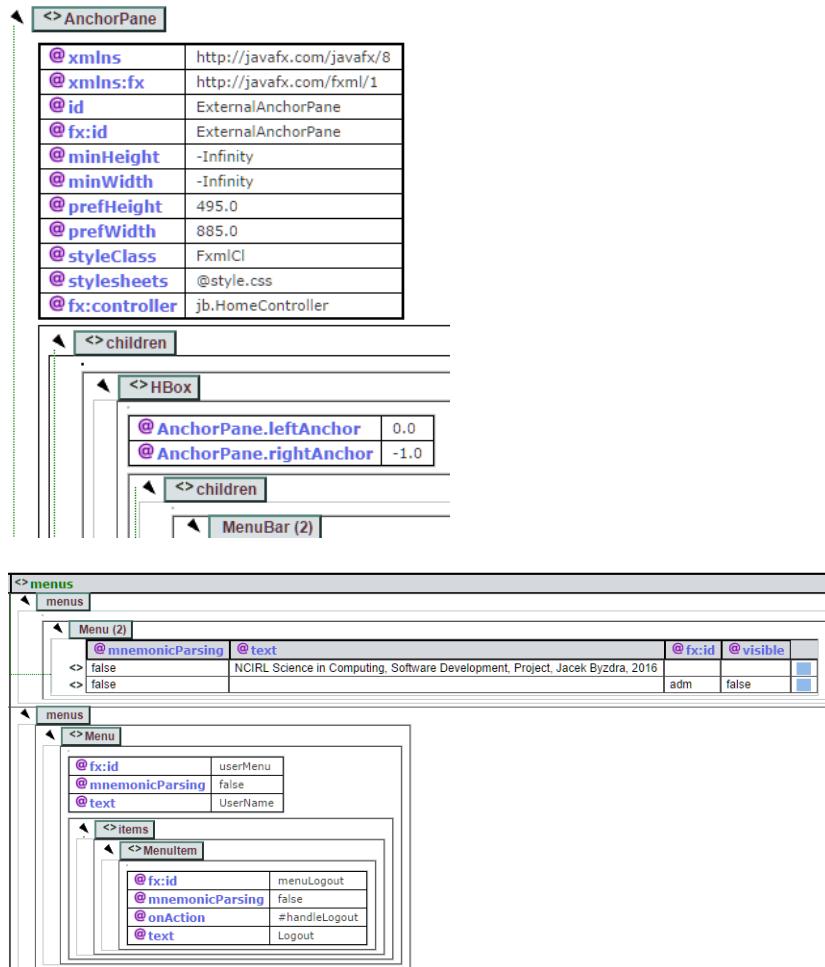


Figure: home.fxml structure

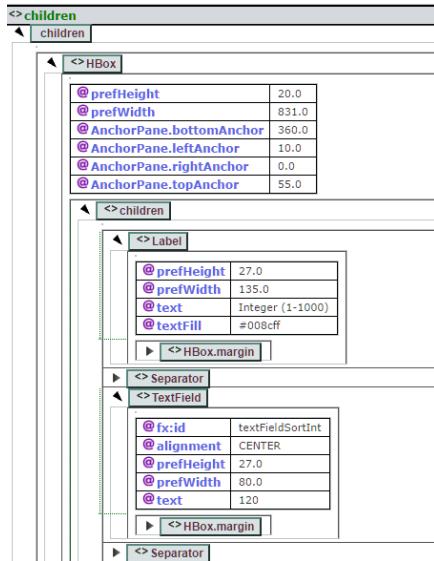


Figure: home.fxml structure cont.

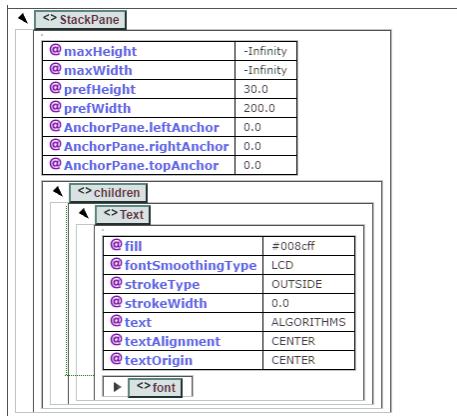
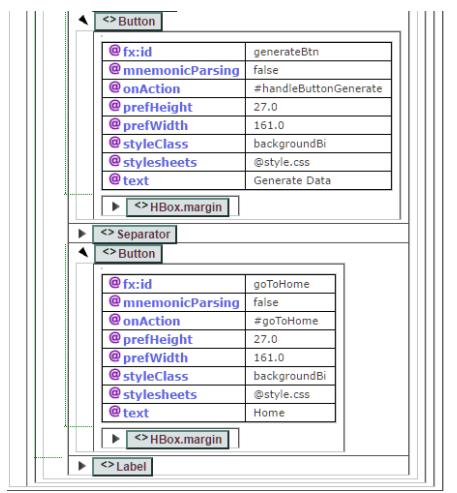


Figure: home.fxml structure cont.

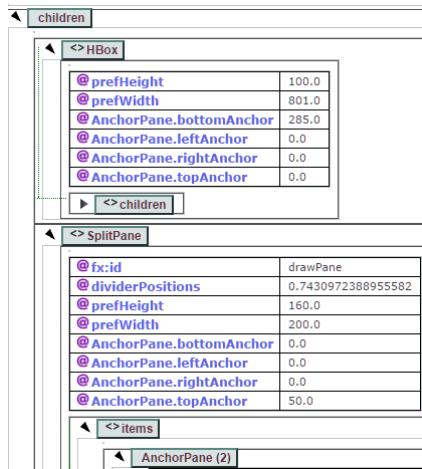


Figure: home.fxml structure cont.

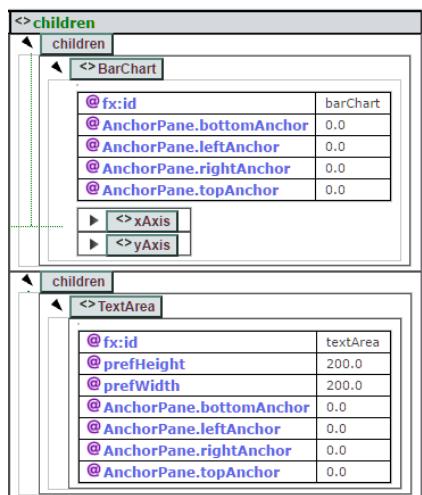


Figure: home.fxml structure cont.

The below figure shows the screen layout of Bars Sorts view

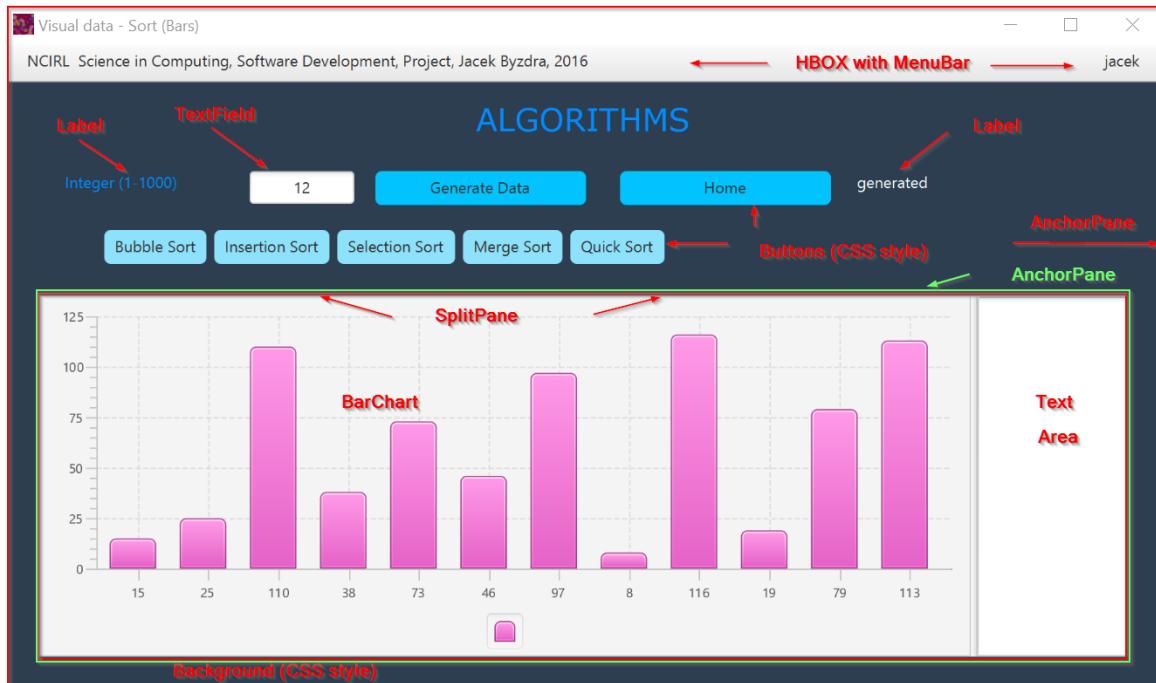


Figure: Bars Sorts view

2.2.6 Design and Architecture – Panels Sorts Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

Sort.java

SortController.java

sort.fxml

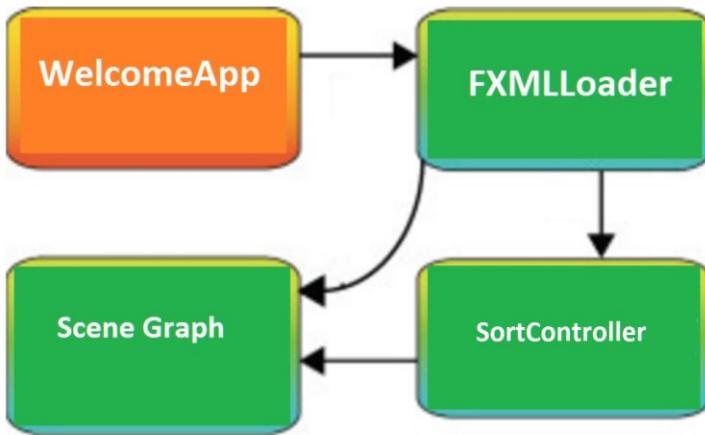


Figure: Structure of JavaFX WelcomeApp Application – Sort Controller

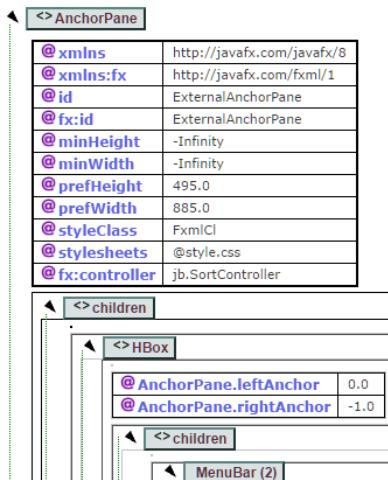
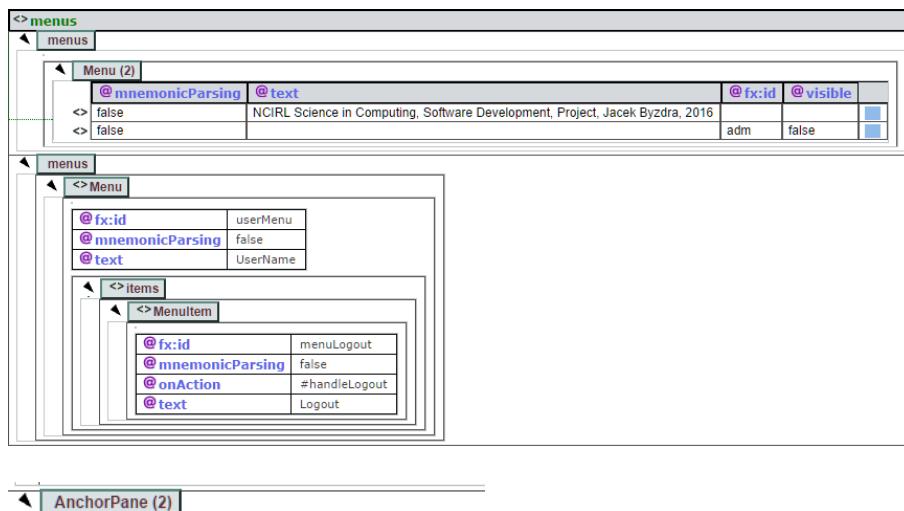


Figure: sort.fxml structure



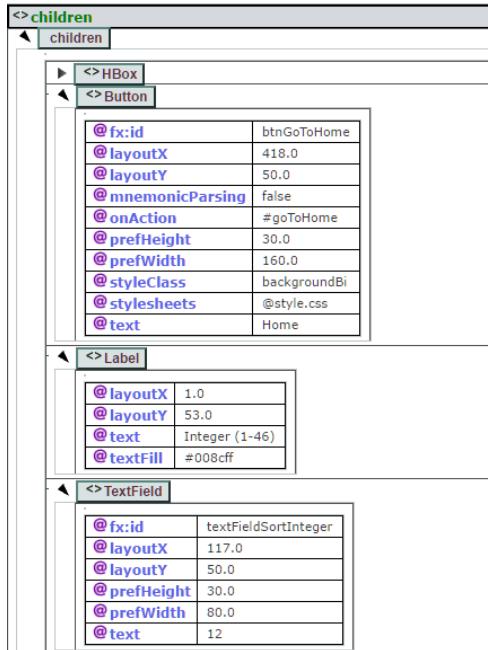


Figure: sort.fxml structure cont.

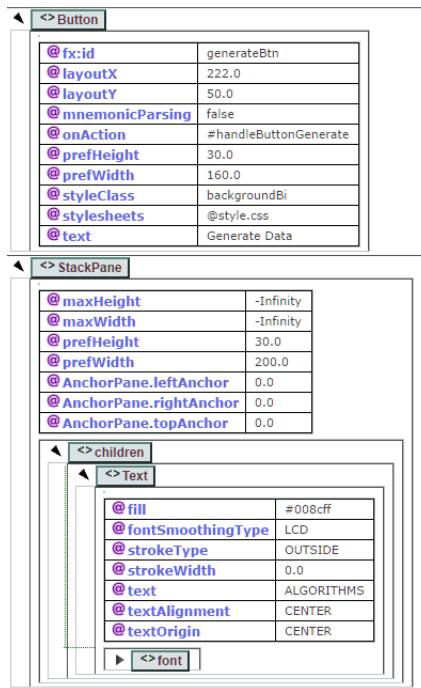


Figure: sort.fxml structure cont

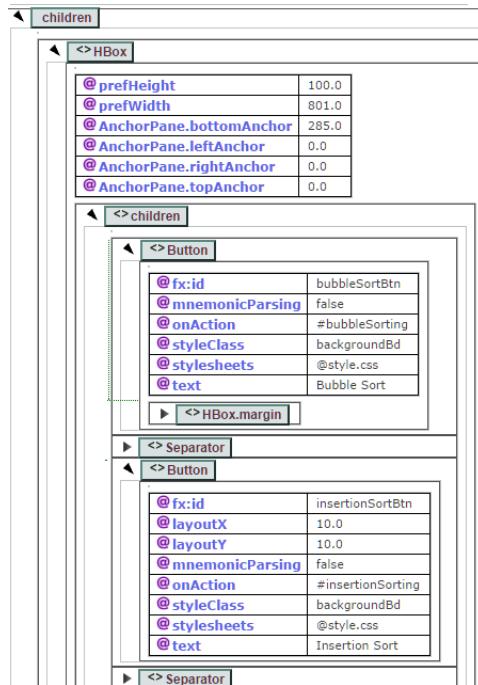


Figure: sort.fxml structure cont

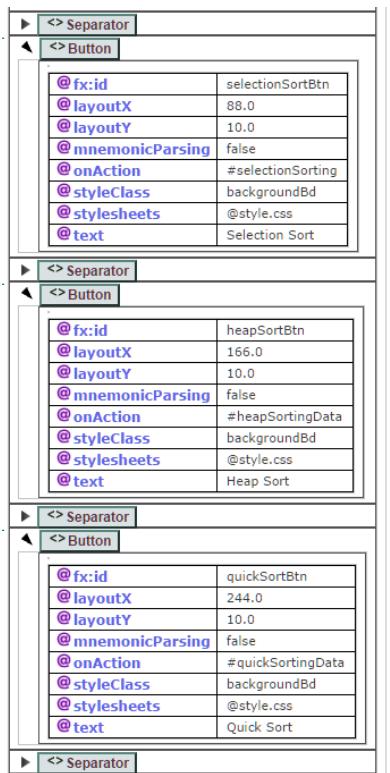


Figure: sort.fxml structure cont

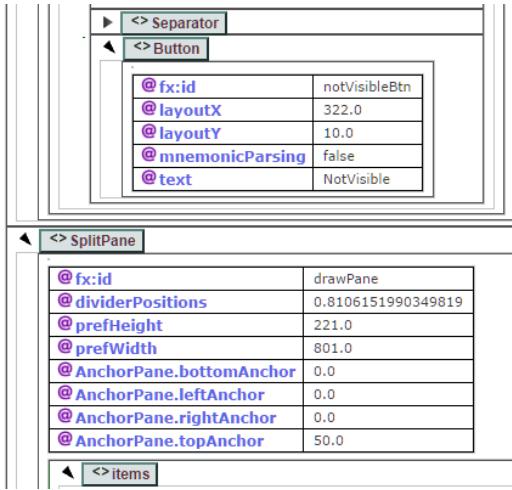


Figure: sort.fxml structure cont.

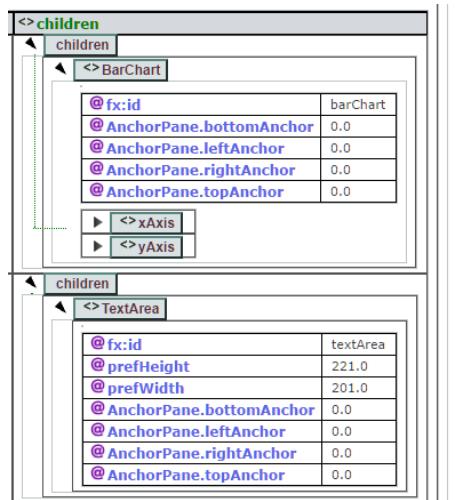


Figure: sort.fxml structure cont.

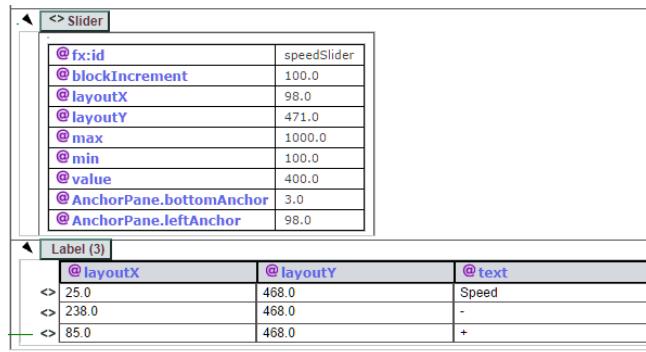


Figure: sort.fxml structure cont.

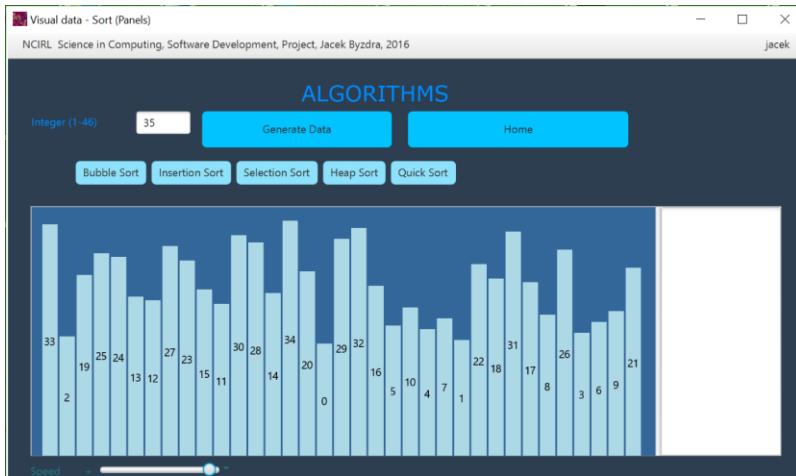


Figure: sort view

2.2.7 Design and Architecture – Tree Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

TreeBSTView.fxml

TreeBSTViewController.java

BST.java

BSTNode.java

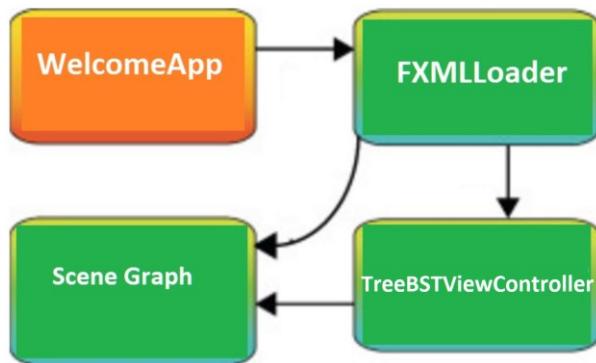


Figure: Structure of JavaFX WelcomeApp Application – Tree Controller

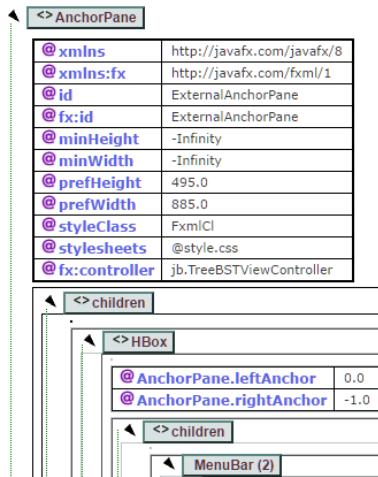


Figure: TreeBSTView.fxml structure

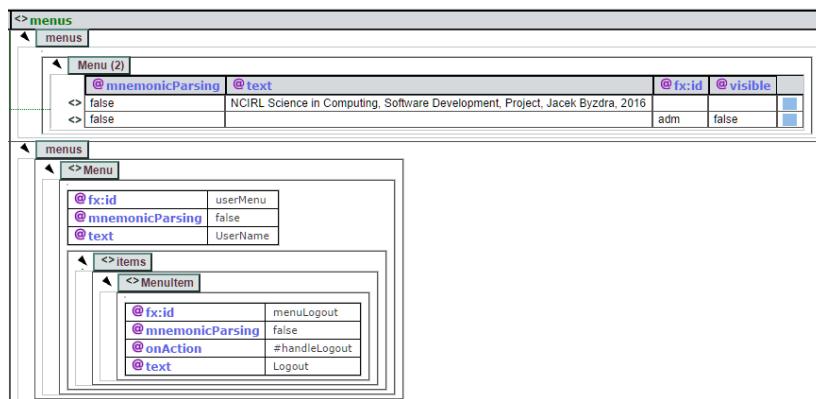


Figure: TreeBSTView.fxml structure cont.

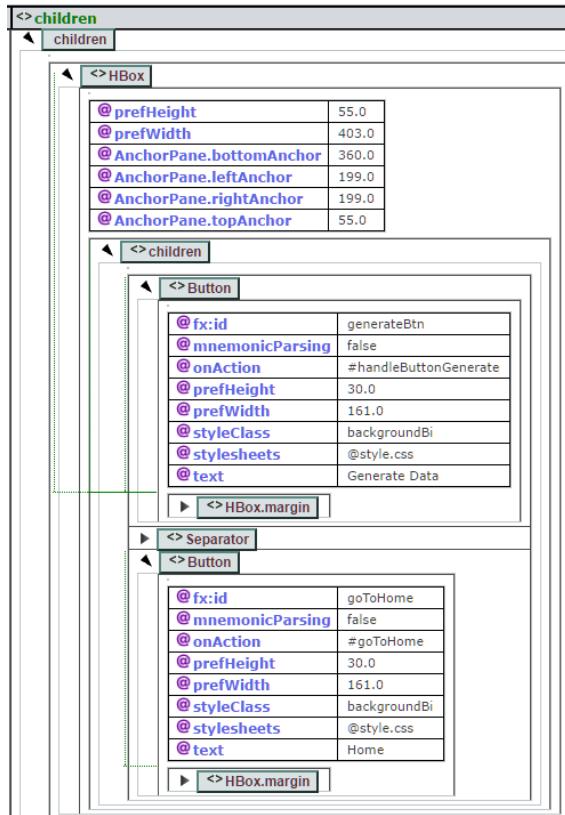


Figure: TreeBSTView.fxml structure cont.

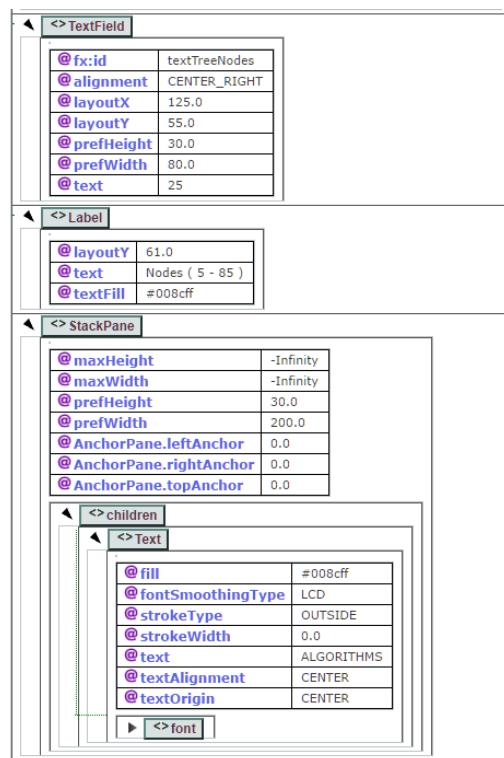


Figure: TreeBSTView.fxml structure cont.

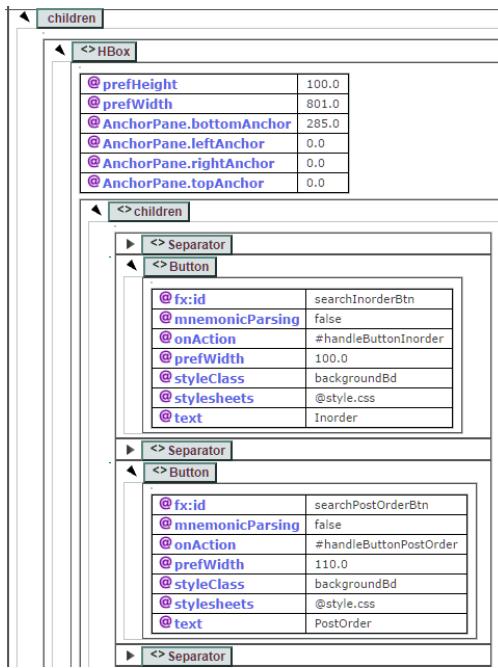


Figure: TreeBSTView.fxml structure cont.

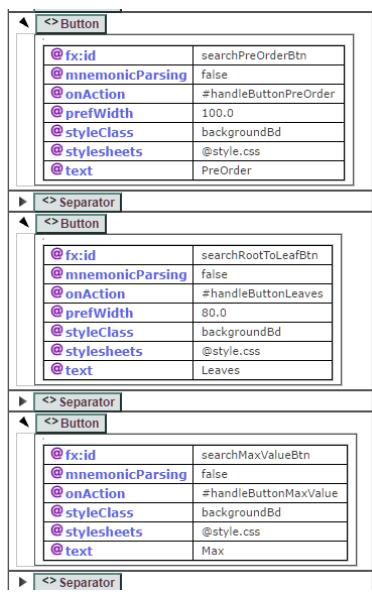


Figure: TreeBSTView.fxml structure cont.

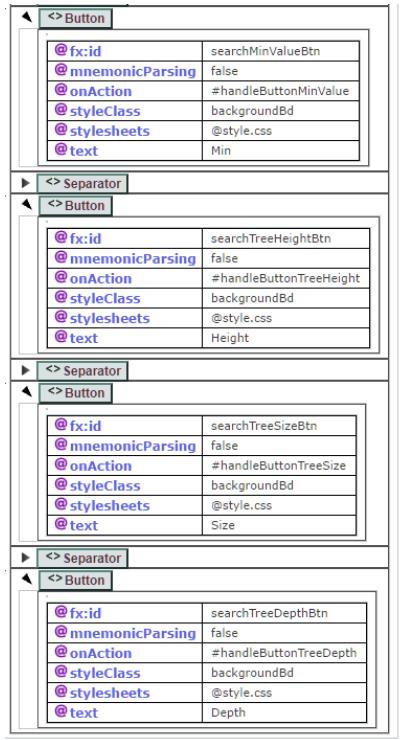


Figure: TreeBSTView.fxml structure cont.

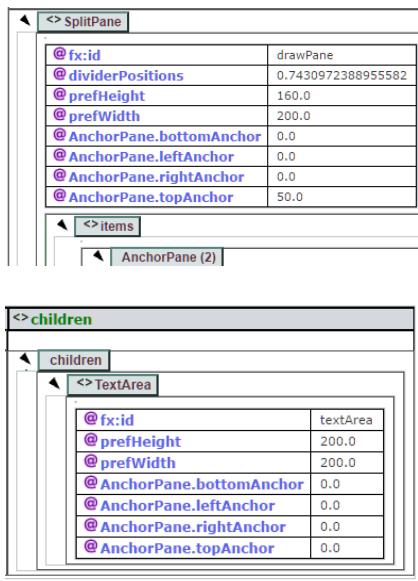


Figure: TreeBSTView.fxml structure cont.

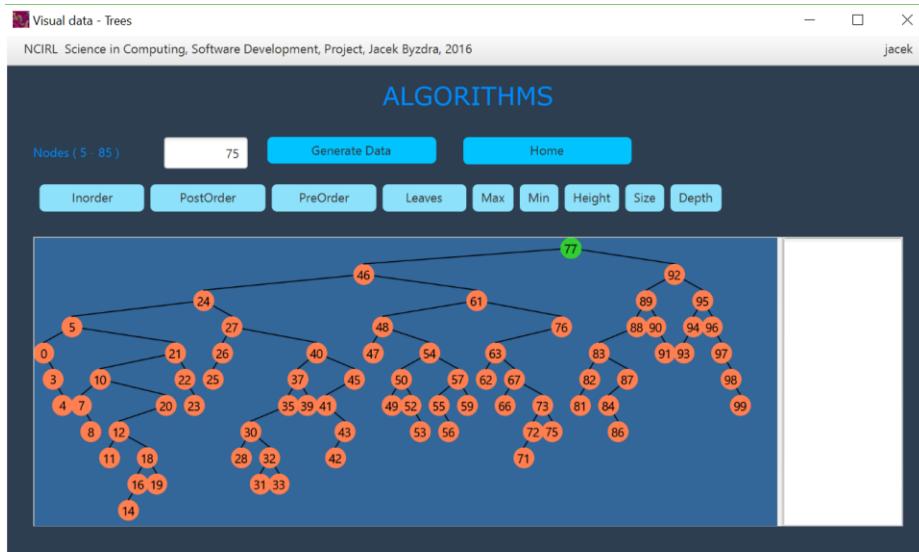


Figure: Tree view

2.2.8 Design and Architecture – Graph Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

Graph.java

GraphController.java

GraphEdge.java

GraphNode.java

graph.fxml

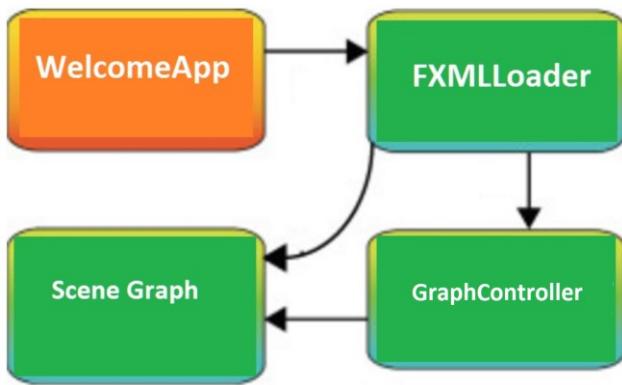


Figure: Structure of JavaFX WelcomeApp Application – Graph Controller

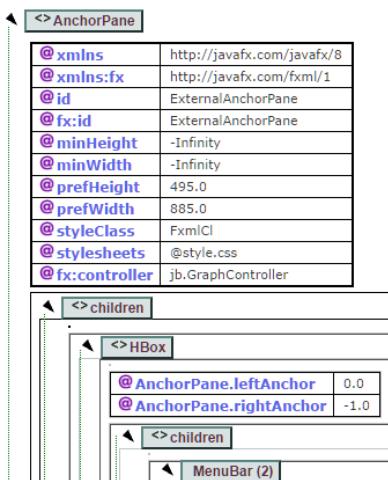
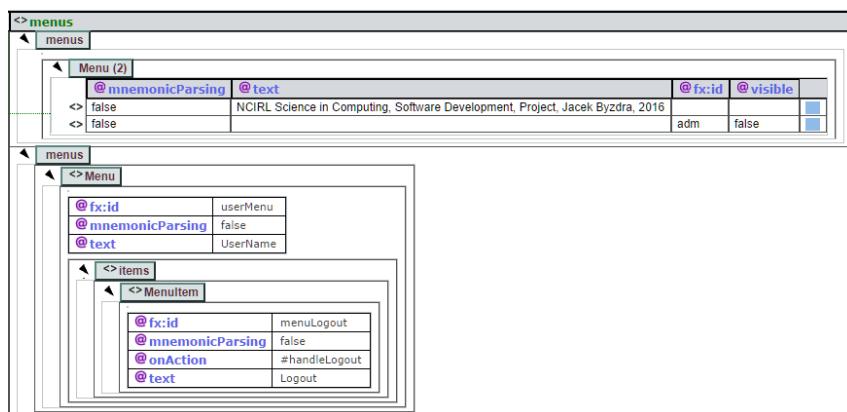


Figure: Graph.fxml structure



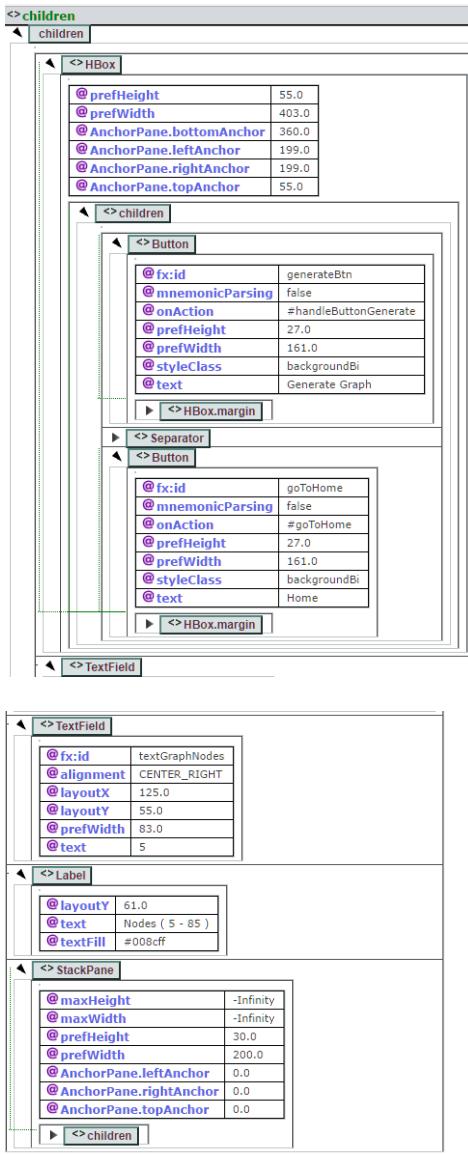


Figure: Graph.fxml structure cont.

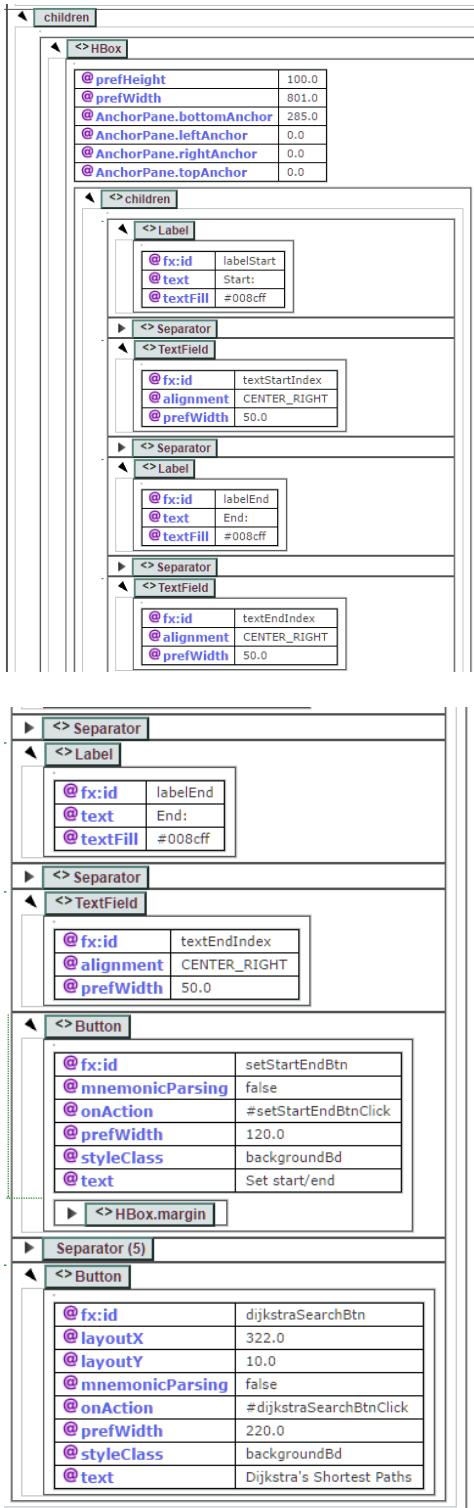


Figure: Graph.fxml structure cont.

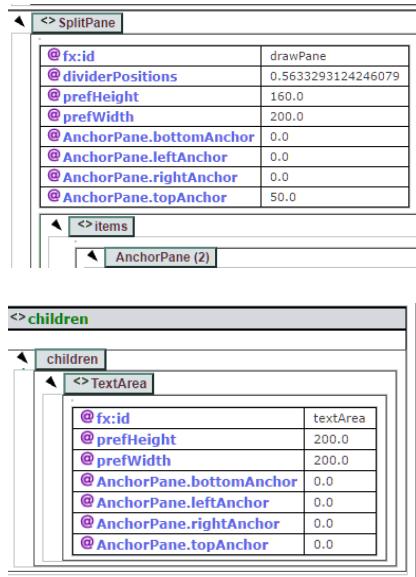


Figure: Graph.fxml structure cont.

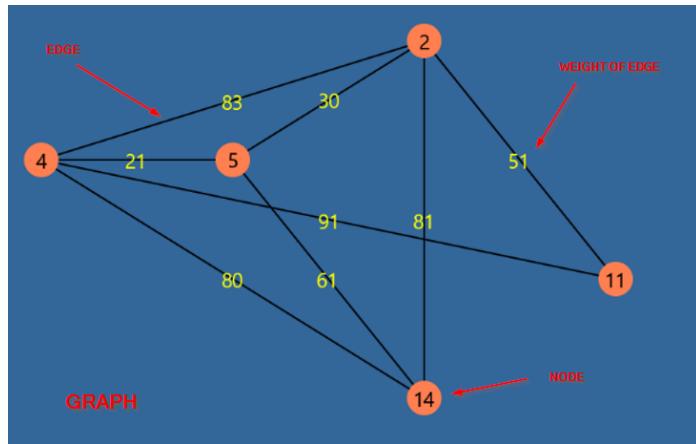


Figure: Graph View

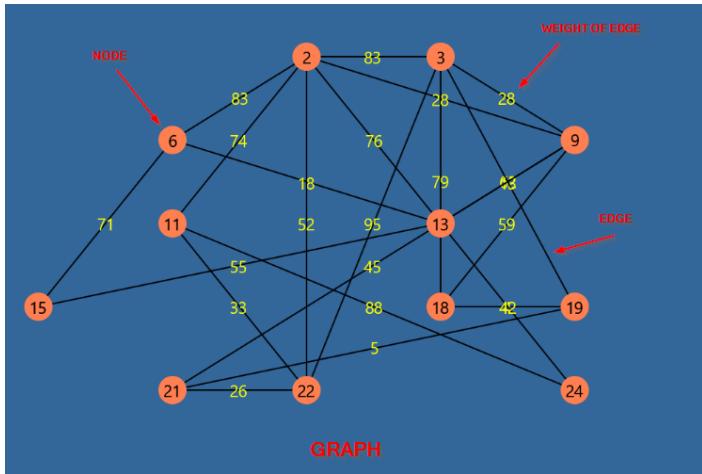


Figure: Graph View cont.

2.3 Implementation

When the user opens application all the use cases are initiated. The functionality of the system, and actions can be performed are clearly defined.

The Welcome view presented below appears as a first view after system is started. The view includes animation effects of title, logo and drawing shapes in the screen. The title changes the color in continuous mode, and the logo moves in the forward and reverse mode in the screen. The circle shapes are randomly generated and drawn, and placed randomly in the screen.

In the center of the window is placed menu Enter button which , when click, allows to go to the next view login.

Below is the printout of Welcome view, and Login view of the system Visual Data.

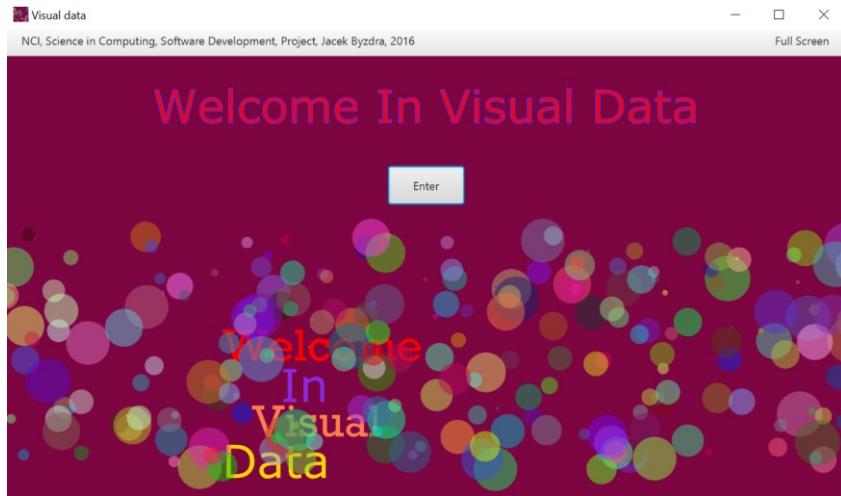


Figure: Welcome view

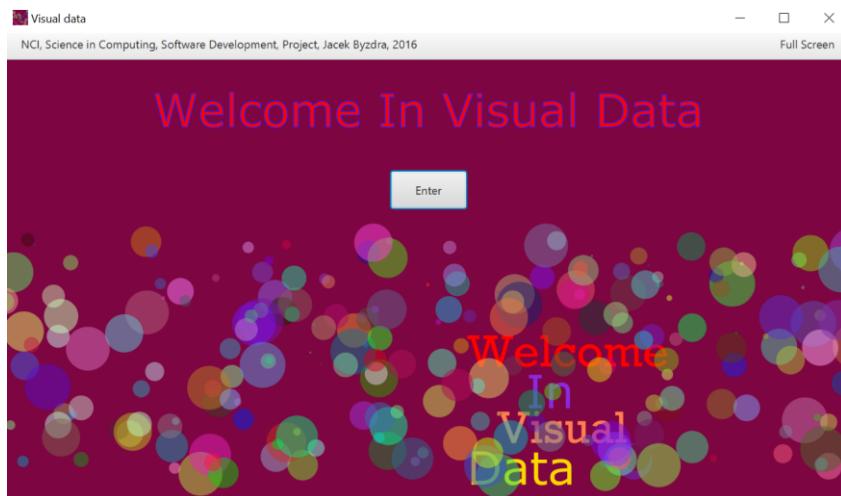


Figure: Welcome view cont.

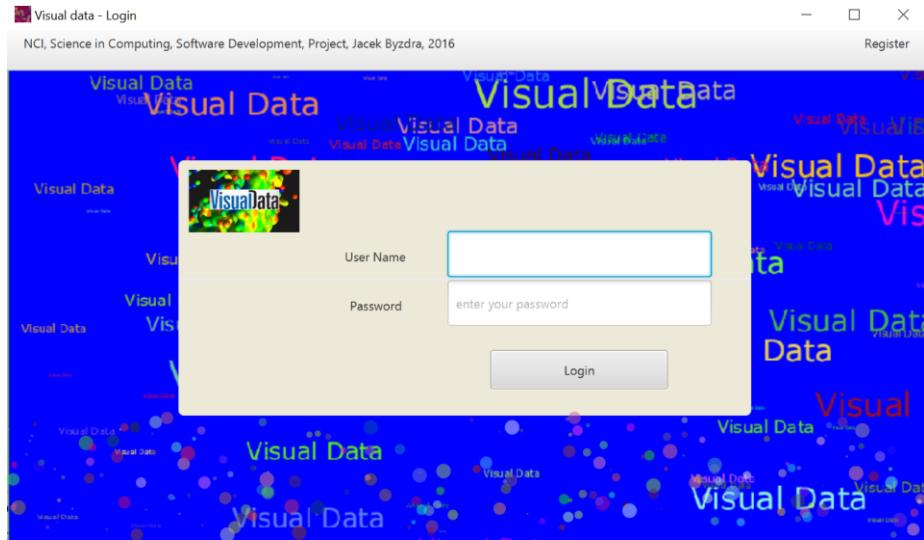


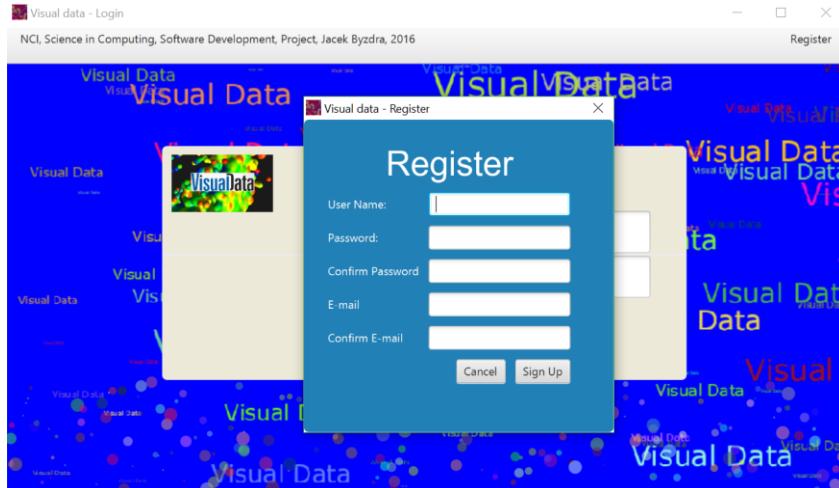
Figure: Login view

When the user fills not correct data to the UserName and/or Password field system will display warning to the user.

System doesn't allow to pass to next level without login to the system.

When the user is not registered in the system he may click the register button in the view of login to process through registration steps.

The printout of Register view is placed below.



In the Register view the system displays several fields in the center of the screen:

User Name – for providing name of the user,

Password – for providing password of the user,

Confirm Password – for confirming password of the user provided in Password field,

E-mail – for providing user email,

Confirm E-mail – for confirming email provided in E-mail field.

In the bottom of the screen are placed buttons Sign Up, and Cancel.

The Sign Up button allows user to start the process of registration when all fields: User Name, Password, Confirm Password, E-mail, Confirm E-mail was provided before. After successful registration to the system the current Register view is switched to Login view , and user may login to the system.

The click of register button invokes methods which switch the login view to register view in the scene.

The Cancel button in Register view, when click, switches back the Register view to Login view.

After successful login to the system the Login view of the system is switched to Menu view.

Below is printout of the Menu view.

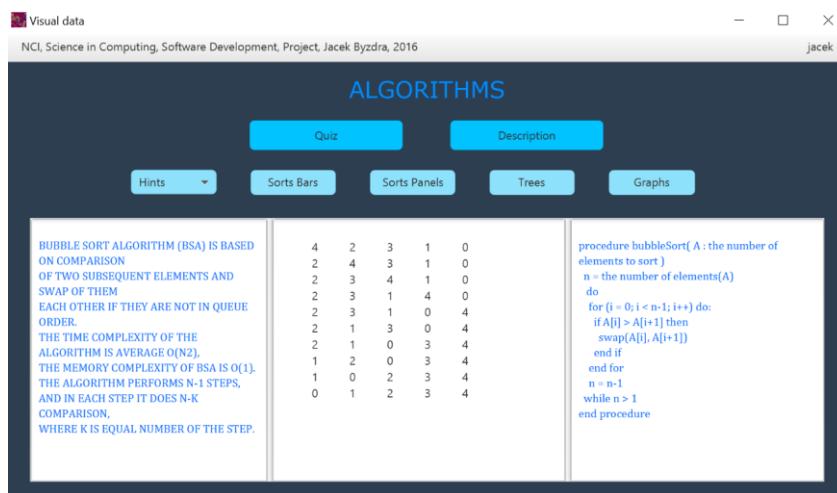


Figure: Menu view

The Menu view presents user several buttons to switchover to different views, and displays hints of the algorithm and data structures in the SplitPane placed in the middle of the screen.

The Button Quiz when pressed switch the current Menu view to Quiz view.

The Button Description when pressed switch the current Menu view to Description view.

The Button Sorts Bars when pressed switch the current Menu view to Sorts Bars view.

The Button Sorts Panels when pressed switch the current Menu view to Sorts Panels view.

The Button Trees when pressed switch the current Menu view to Trees view.

The Button Graphs when pressed switch the current Menu view to Graphs view.

The Menu Hints when pressed open several Menu items to mark to display hints on different subjects.

Below is printout of Menu view when Hints button is pressed.

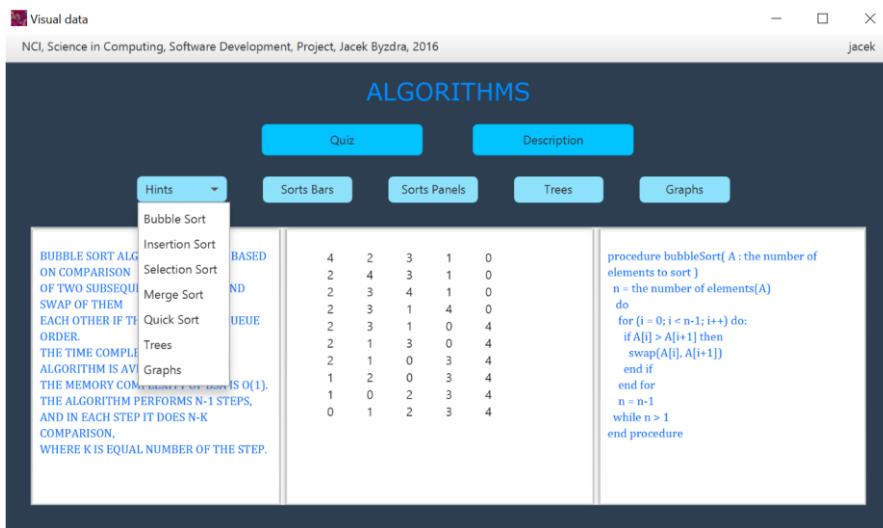


Figure: Menu view when Hints Menu is pressed

In the Hints Menu system displays: BubbleSort, InsertionSort, SelectionSort, MergeSort, QuickSort, Trees, and Graphs hint Menu Items. When user marks

one of the Menu Items system displays the information hints about marked subject. The information includes basic information and pseudo code of the algorithm or data structure .

When user press Quiz button in Menu view system switch the Menu view to Quiz view.

Below is printed Quiz view start page.

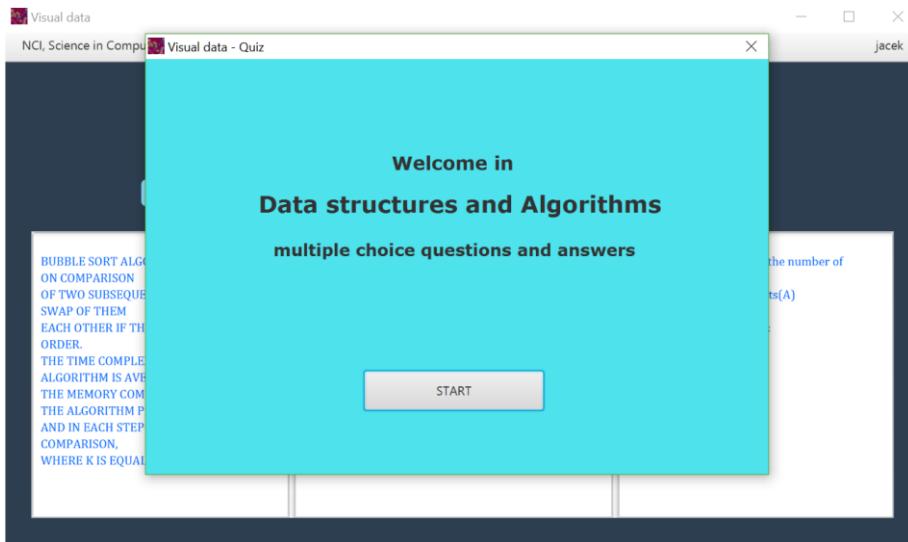


Figure: Quiz view start page

In the start page of the Quiz view system displays the user information about quiz and startbutton.

The start button when is pressed switch the Quiz view to the new stage when question and answers are displayed to the user.

Below is printout of the

The new stage of the Quiz view when start button is pressed is displayed below.

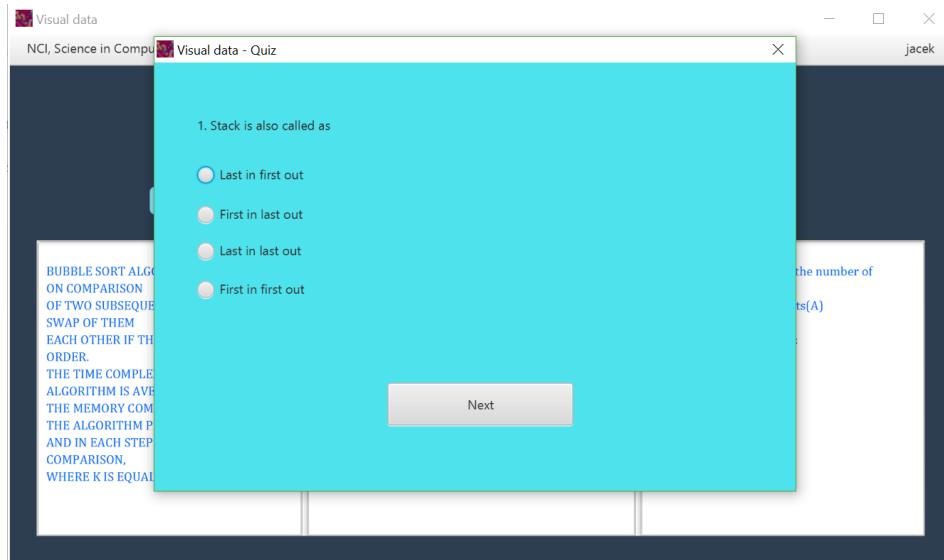


Figure: Quiz view new stage

The new stage display question and multiple answers to the user. User may mark only one right question and press next button to go the next step where new questions and answers are displayed in the screen.

The quiz includes 5 steps where question and answers are displayed to the user.

Each time user starts new quiz new questions and answers are generated from the database. When user pass all 5 steps in the quiz the system display results of the quiz to the user.

Below is example of the printout of the quiz results view.

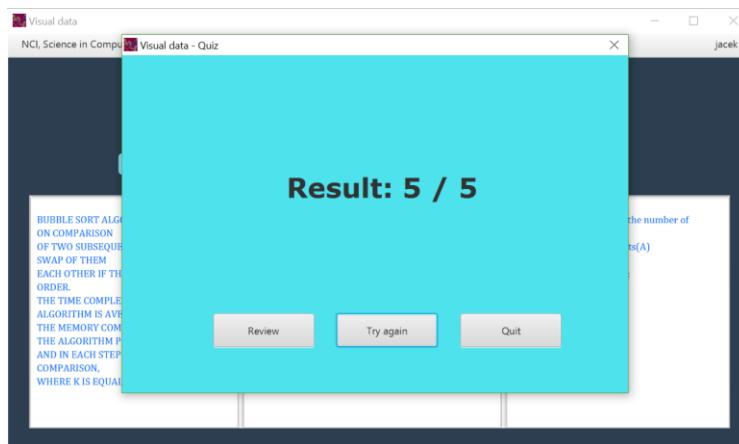


Figure: Example of quiz results view

In the results view of the quiz system displays the received results and buttons: Review, Try again, Quit.

The Button Review displays step by step the detailed results of answered questions. The right question is marked with a green cross, and wrong answer is marked with a red cross.

Below is example of the detailed results on answered question.

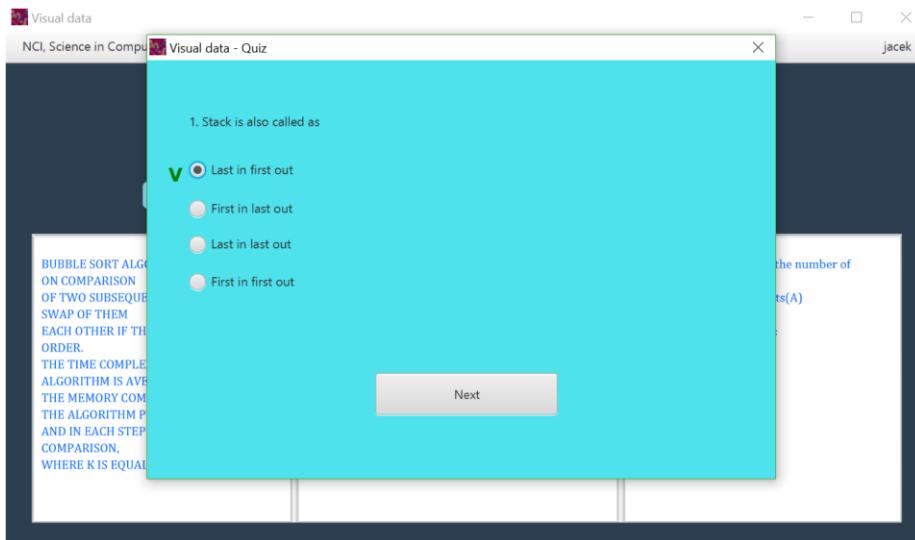


Figure: Example of the detailed results on answered question.

When user press Try Again button in the quiz results view system generates new questions and answers , and starts new quiz.

When user press Quit button in the quiz results view system switch the quiz view back to the Menu view.

When user press the button Sorts Bars in Menu view system switch the current Menu view to the Sorts Bars view.

When user press the button Sorts Panels in Menu view system switch the current Menu view to the Sorts Panels view.

When user press the button Trees in Menu view system switch the current Menu view to the Trees view.

When user press the button Graphs in Menu view system switch the current Menu view to the Graphs view.

When user press the button Description in Menu view system switch the current Menu view to the Description view.

Below are several printouts of views which appears when user press button in the Menu view.

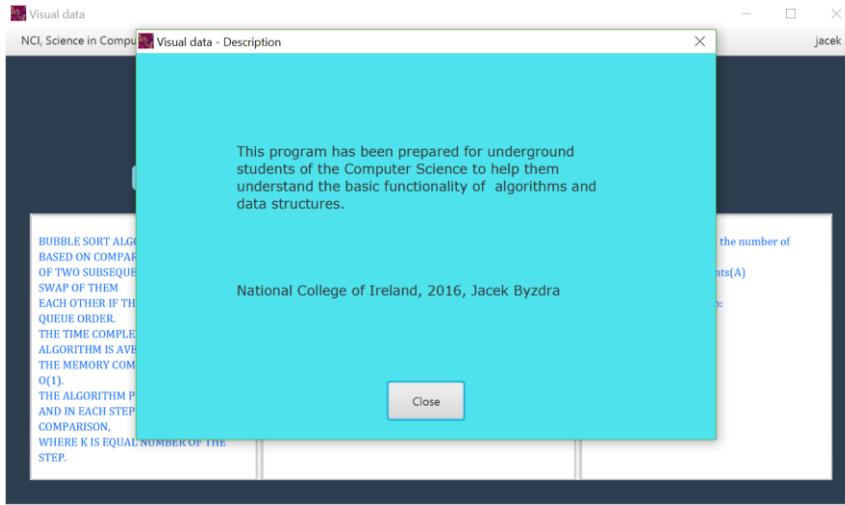


Figure: Description view

In the Description view system displays information about Visual Data system.

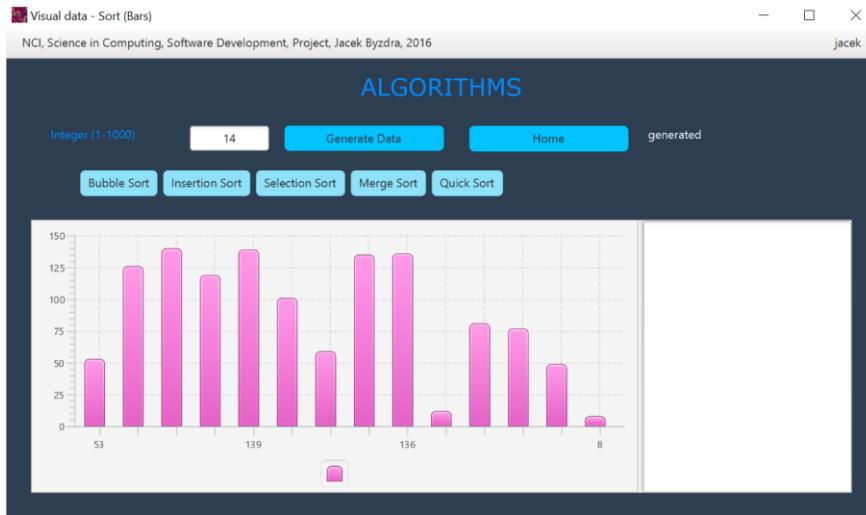


Figure: Sorts Bars view

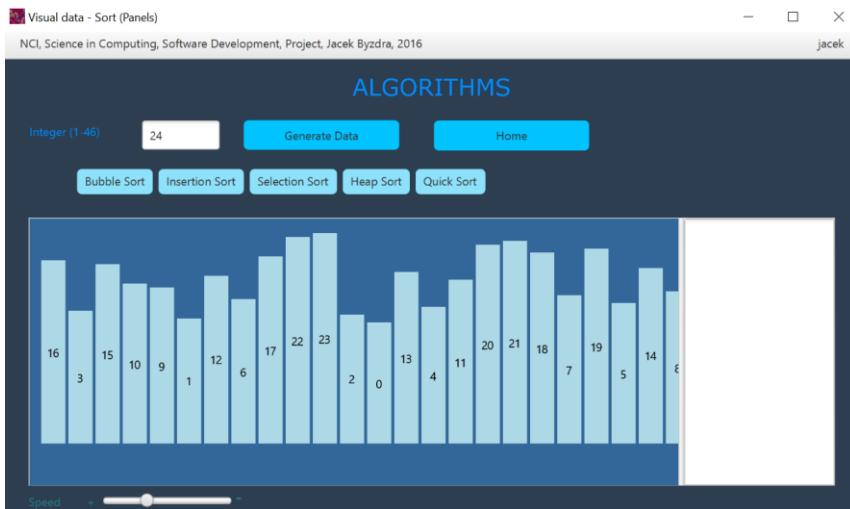


Figure: Sorts Panels view

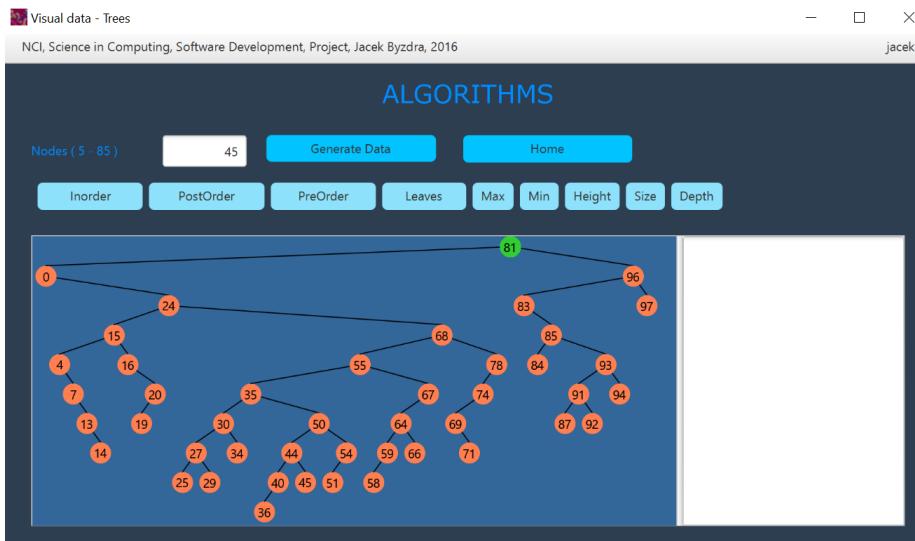


Figure: Trees view

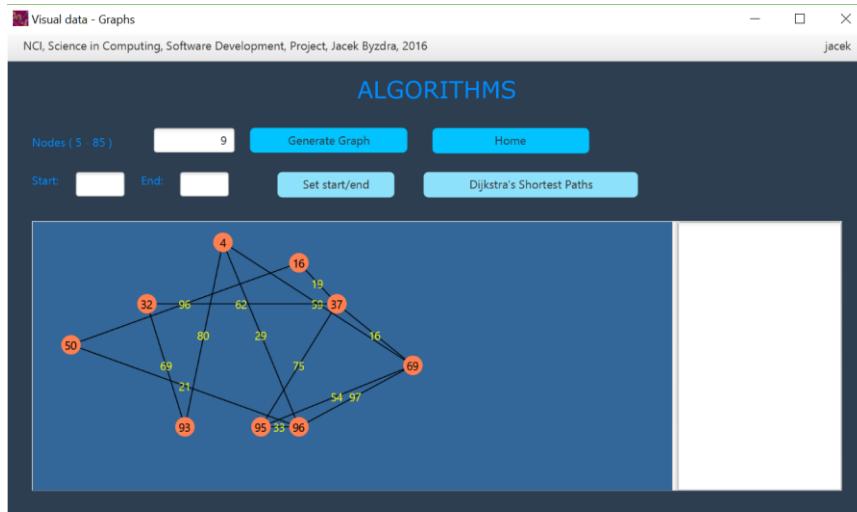


Figure: Graphs view

In the Sorts Bars view system displays several buttons for user to operate on algorithms.

The Button Generate Data, when the correct value of the number of the bars was provided to the search label before, when pressed will generate randomly the values of the Bars.

The integer number of the Bars in Sorts Bars view is limited in the range from 1 to 1000 . This was limited to show the visualization in clear mode, not displaying too much Bars, or too less in the screen .

The Button Generate Data will invoke generation of new data for searching algorithm and new bars will be displayed in bars chart view in the Split Pane in the middle of the screen.

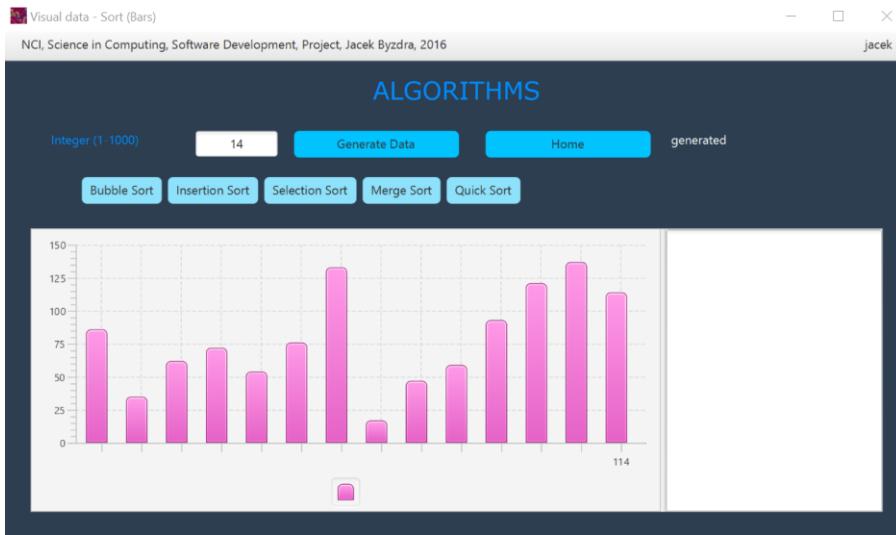


Figure: Sorts Bars view

The Button Home when pressed will switch the current Sorts Bars view to the Menu view.

The Buttons Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, and Quick Sort when pressed invoke start of searching algorithm and start of measuring of time of it performance.

When the search algorithm ends the search system will display solution of the searching showing soted bars in the SplitPane view.

System also display the time of the algorithm performance when the search ends.

Below is printout from searching of the algorithms in Sorts Bars view.

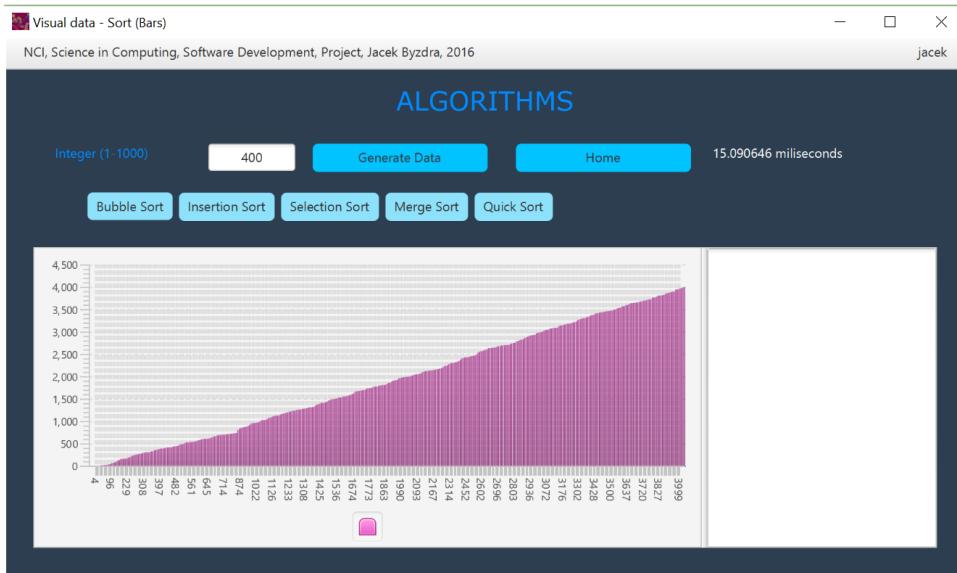


Figure: Sorts Bars view – invoked Bubble Sort algorithm

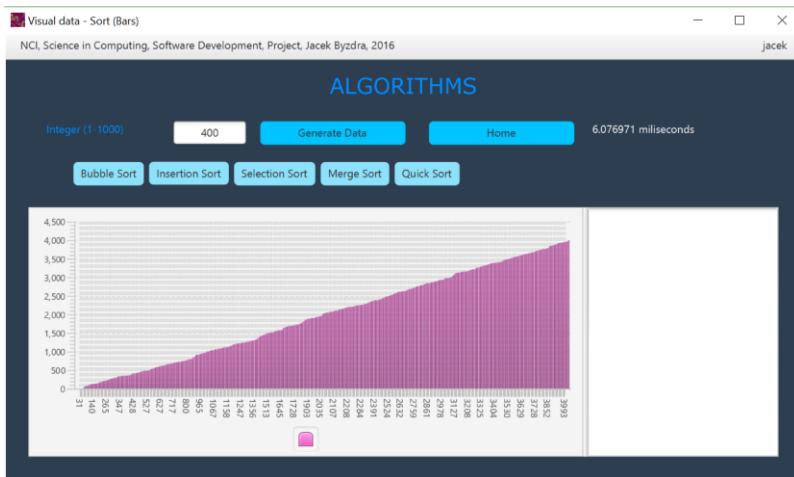


Figure: Sorts Bars view – invoked Insertion Sort algorithm

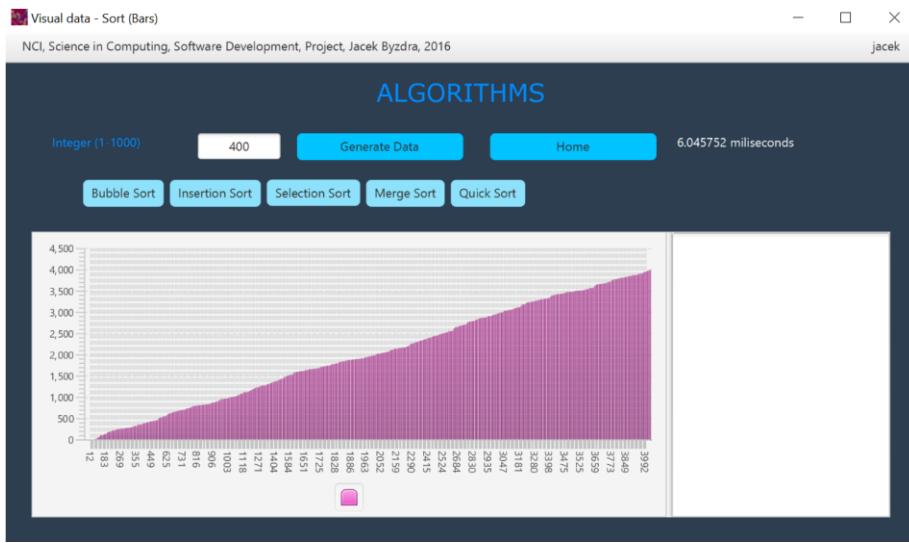


Figure: Sorts Bars view – invoked Selection Sort algorithm

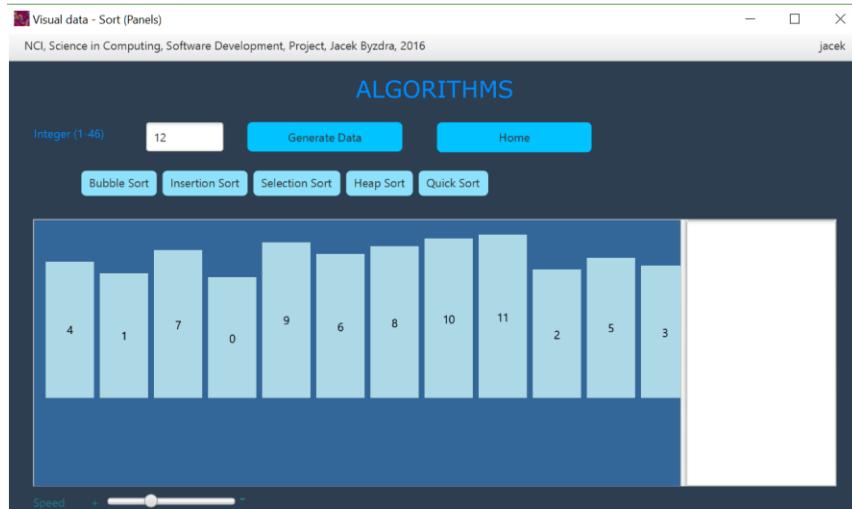


Figure: Sorts Bars view – invoked Merge Sort algorithm



Figure: Sorts Bars view – invoked Quick Sort algorithm

When user press button Sorts Panel in Menu view system will switch the current Menu view to Sorts Panel view.



In the Sorts Panels view system displays several buttons for the user to operate on algorithms and view of sort panels in the Split Pane view, in the middle of the screen.

The Button Generate Data, when the correct value of the number of the panels was provided to the search label before, when pressed will generate randomly the values of the Panels.

The integer number of the Panels in Sorts Panels view is limited in the range from 1 to 46 . This was limited to show the visualization in clear mode, not displaying too much Panels, or too less in the screen .

When the user press button : Bubble Sort, Insertion Sort, Selection Sort, Heap Sort, Quick Sort system will starts the searching algorithm.

When the user press Home button system will switch the current Sorts Panels view to the Menu view.

When the user change the position of the slider before starting searching of the algorithm system will change the rate of the searching speed.

Below is example of the Bubble Sort search printout in Sorts Panels view.

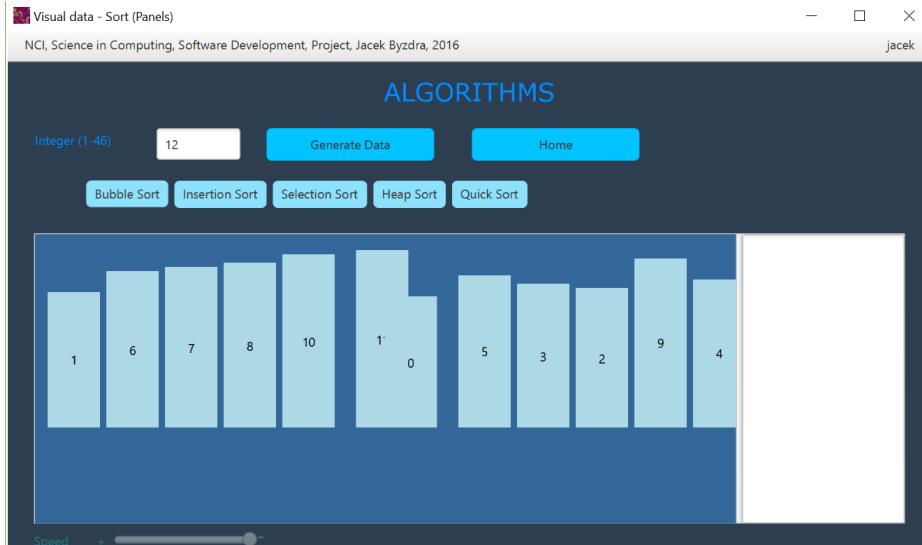


Figure: Sorts Panels view – invoked Bubble Sort algorithm in low speed

The Sorts Panels view in comparison to Sorts Bars view displays the moving panels in the Split Pane view. When the user start the searching method the panel /panels depends on searching method, moves in horizontal direction and swap another panel in the Split Pane view. The swap movement of the panels is animated until the search algorithm sorts all the panel in increased order.

Each panel has values assigned to it , which was generated by user when Generate Data was pressed.

This animation of Sorts Panels shows in details which panel is swapped showing the algorithm's steps.

When the user press Trees button in Menu view system will switch the current Menu view to Trees view.

Below is printout of the Trees view.

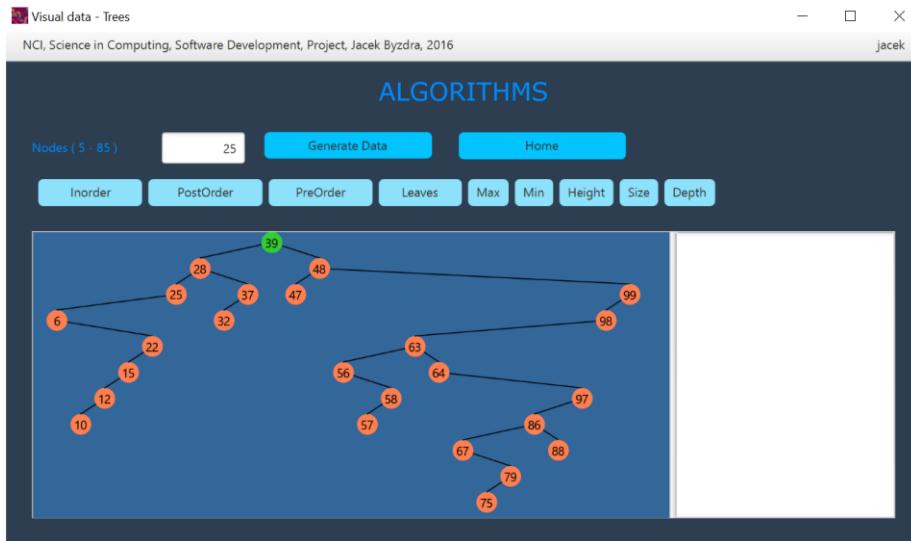


Figure: Trees view

In the Trees view system displaysd severals buttons to operate on Binary Search Tree and generated tree in the Split Pane view , in the cntere of the screen.

The Button Home when pressed switch the current Tree view to Menu view.

The Button Generate Data, when the correct value of the number of the tree nodes was provided to the Trees label before, will generate randomly the values of the Nodes and display Binary Search Tree from generated Nodes values.

The integer number of the Nodes in Binary Search Tree is limited in the range from 5 to 85 . This was limited to show the visualization in clear mode, not displaying too much Nodes, or too less in the screen .

In the Trees view user may press button: Inorder, PostOrder, PreOrder, Leaves, Max, Min, Height, Size, Depth to perform operation on Binary Search Tree.

The buttons invoke searching method on the tree generated before by user. The searching methods are animated showing solution of the search in different color and displaying the textual solution in TextArea in in the Split Panel on the screen.

The root Node of the Binary Search tree is displayed in green color.

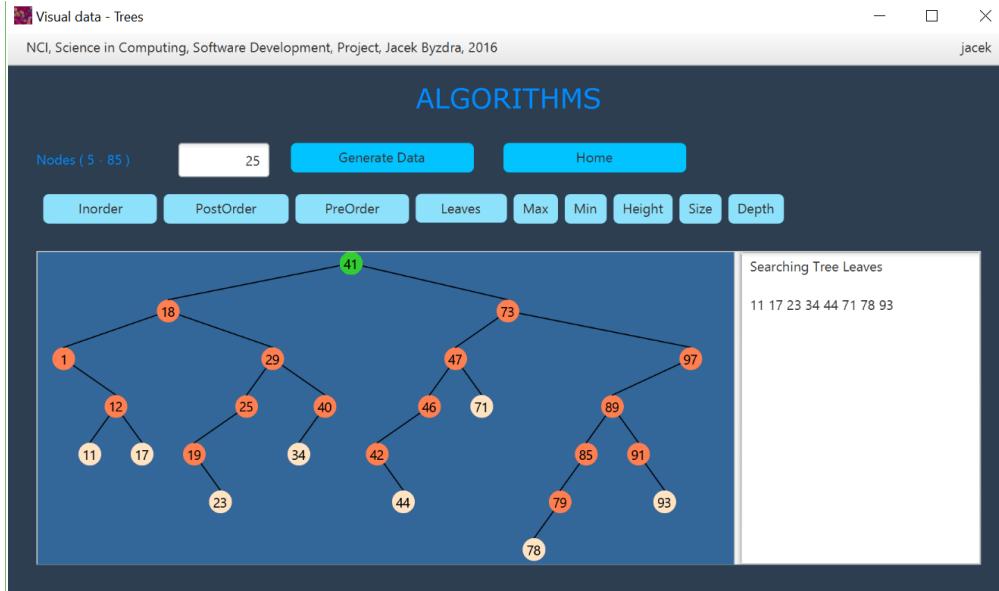


Figure: Trees view – invoked algorithm search Leaves in the BST tree

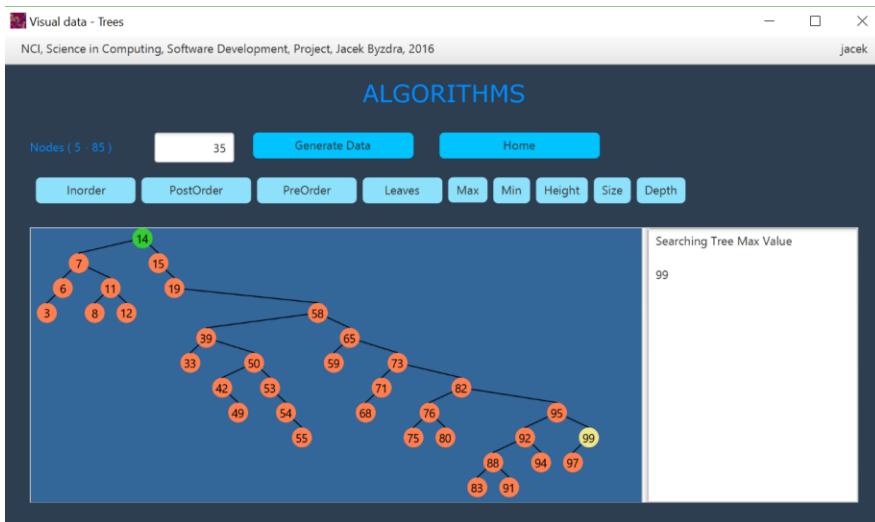


Figure: Trees view – invoked algorithm Max value Node

When the user press the button Graphs in the Menu view system will switch the current Menu view to Graphs view.

Below is an example of the Graphs view printout

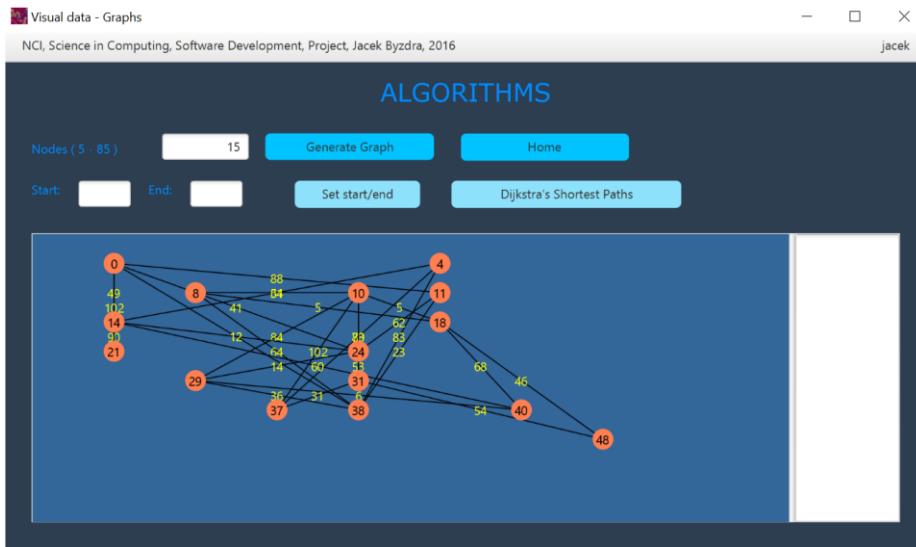


Figure: Graphs view

In the Graphs view system displays several buttons to operate on the graph.

The Home button when pressed will switch the current Graphs view to the Menu view.

The Button Generate Graph, when the correct value of the number of the Nodes was provided to the Graph label before, will generate randomly the values of the Nodes and display Graph from generated Nodes values.

The integer number of the Nodes in Graph is limited in the range from 5 to 85 . This was limited to show the visualization in clear mode, not displaying too much Nodes, or too less in the screen .

The Start Label defines the start point of the searching algorithm in the Graphs view.

The End Label defines the end point of the searching algorithm in the Graphs view.

The Start start/end button will set the start and end Node in the graph, if the start/end nodes were defined by user before.

The start/end point Node will be colored in violet after start/end Node is set.

The Dijkstra's Shortest Paths button when pressed, and start/end Node was set before, will search and find the solution for the shortest paths way according to the Dijkstra's algorithm.

The nodes and edges on the shortest path way will be displayed in Violet color when the solution is found. The solution of the searching is also displayed in the TextArea in the Split Panel in the screen.

Below is an example of the printout of Shortest path way invoked when Dijkstra's Shortest Path button was pressed.

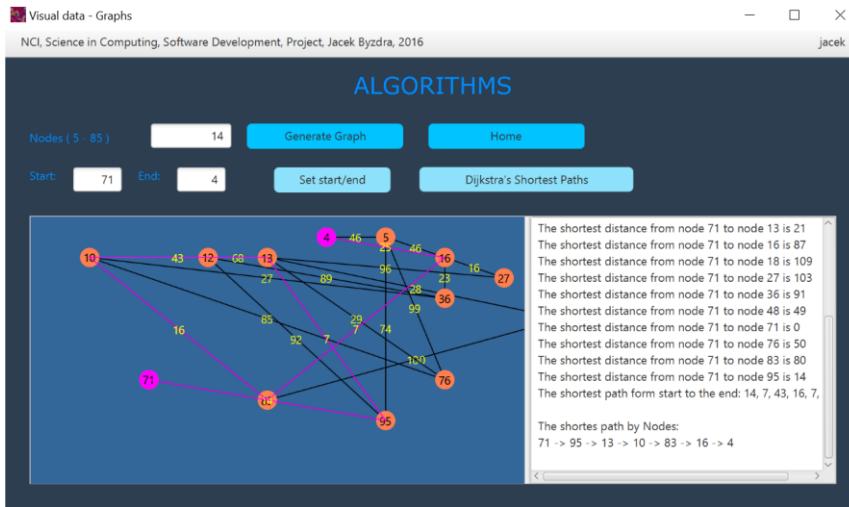


Figure: Graphs view – invoked Dijkstra's Shortest Path algorithm

2.3.1 Implementation – main algorithm presentation – graph

Main algorithms in application - Graph presentation.

Graph is created dynamically.

First user enters number of the graph nodes he wants to generate. The value must be in range 5-85 nodes. Creating graph is based on a matrix.

As a maximum of nodes is 85 then maximum matrix size is defined as 10.

The Size 10 allows use matrix $10 \times 10 = 100$ items, and it is enough to cover all 85 nodes.

The number of items in the matrix will be always \geq (more or equals) than the number of nodes, so we can random position each node in the matrix. This ensure each time we generate a graph, it will have different sequences of nodes in matrix. The matrix is computed according to the screen size. Each node keeps its position after it is located in the matrix.

In next steps we generate edges.

As default each the node in the matrix has connection with all other nodes.

It makes a graph unreadable, especially when there are many nodes..

To ensure dynamic generation of the graph, and to ensure readability of the graph the program eliminates some edges randomly, in such a way that each node in the graph has no more than 3 connections to others.

This solution ensures that even for the same number of nodes the graph view is readable , and each time the graph is different.

I developped below methods to create graph :

In GraphController.java class methods:

- createGraph()
- randomMatrixSize(int numberGraphNodes)

In Graph.java class methods:

- createRandomGraphNodes(int nodeNumbers, int matrixSize)
- createEdgesGraphNodes()

Finally the graph presentation is finished by GraphControll method: DrawGraph().

Drawing graph is based on the position which is kept by each node.

The Node position refers to the screen coordinates. The method consists of two next methods: DrawNode(), and DrawEdge(), and it is based on drawing graphical object: circle, screen and text on the screen position. In details the objects are children of the container AnchorPane which is located in JavaFX scene.

2.3.2 Implementation – main algorithm presentation – sorting algorithm with animation

The common sorting algorithms used in the program has been enhanced with graphical presentation. The number of data to sort is entered by the user. The data are generated randomly.

The graphical objects – rectangles are drawn in animation panel.

The height of each panel - rectangle corresponds to the randomly selected number. The width of the panels depends on the number of data to fit on the screen.

The sorting methods receive as input data to sort. It is a list of numbers to draw and wrap the graphic objects. The Sorting algorithms operates on numbers.

The corresponding graphic objects are set in the same order. Each sort method returns an object 'SequentialTransition'.

The SequentialTransition allows to register each stages of sorting and then recreate the position of rectangles.

There is possible to set a speed of animation in the program.

The sorting methods are part of class Sort.java, and following methods were created:

- bubbleSort()
- insertionSort()
- selectionSort()
- quickSort()
- heapSort().

2.3.3 Implementation – main algorithm presentation – Binary Search Tree presentation

The graphical presentation of the tree is linked with the algorithms building and searching the binary tree.

TreeBSTViewController.java class is responsible for generating and drawing a tree.

The user specifies the number of vertices of the tree. The range of number of nodes is five to 85 and their values are generated randomly. Class BST.java is responsible for the creation of the Binary Search Tree . In the class were created the methods for creating and searching BST tree.

The tree is arranged in the AnchorPane container symmetrically. Each node tree keeps its value and position location on the screen. The Method DrawTree () draws graphics lines and circles with appropriate description text for the value.

The tree is drawn recursively, first left part of the tree is drawn then the right part.

2.3.4 Implementation – class DBConnection.java

public class DBConnection :

- class responsible for connection to database MySQL
- method: getConnection() : return connection
- method: connect() : set up connection ,
- method: validateUser(String user, String password) : validation of user database,
- method: registerUser(User user) : insert user data into database
- method: getQuizQuestions() : return all quiz questions from database, read only questions ,
- method: getQuestionAnswers(QuizQuestion question): return answers belonging to quiz question.

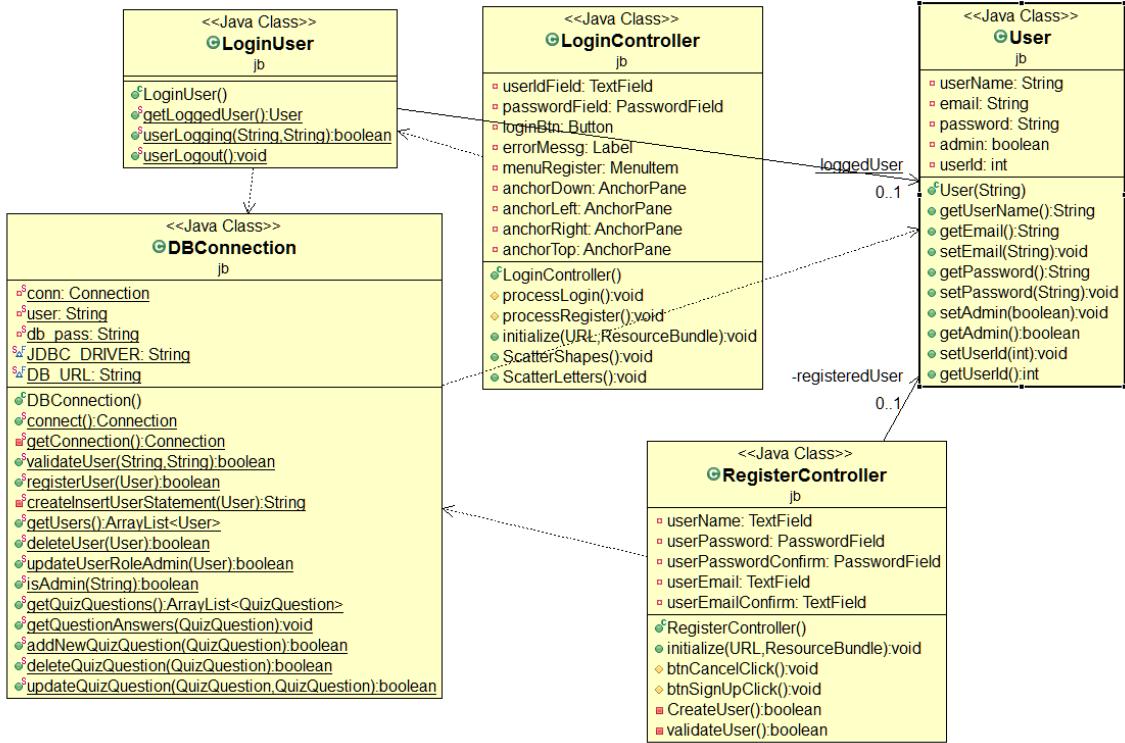


Figure: DBConnection.java class

2.3.5 Implementation – class WelcomeApp.java

public class WelcomeApp extends Application:

- Main class of application. This class manages stages and scenes in JavaFX. It also switches to different stages, pass control to different JavaFX controllers.
- method: public static WelcomeApp getInstance() : return instance of Class (used in all controllers)
- method: public void start(Stage primaryStage) throws Exception { : to init primary stages
- method: public void gotoWelcome() : go to Welcome scene

- method: public void gotoHome() : go to Home scene
- method: public void gotoSort() : go to Sort scene
- method: public void gotoMenu() : go to Menu scene
- method: gotoLogin() : go to Login scene
- method: gotoTreeBST() : go to TreeBST scene
- method: gotoGraph() : go to Graph scene
- method: public void gotoRegister() : open register window by swapping stages
- method: closeRegister() : close stage Register
- method: gotoDescription() : open description window swapping stage
- method: closeDescription() : close description window swapping stage
- method: gotoQuiz() : open quiz window swapping stage
- method: closeQuiz() : close quiz window swapping stage
- method: replaceSceneContent(String fxml) : replace content of scene in primaryStage
- method: showMessage(String title, String header, String content) : display message for user in all application. Keeps uniform format of the message

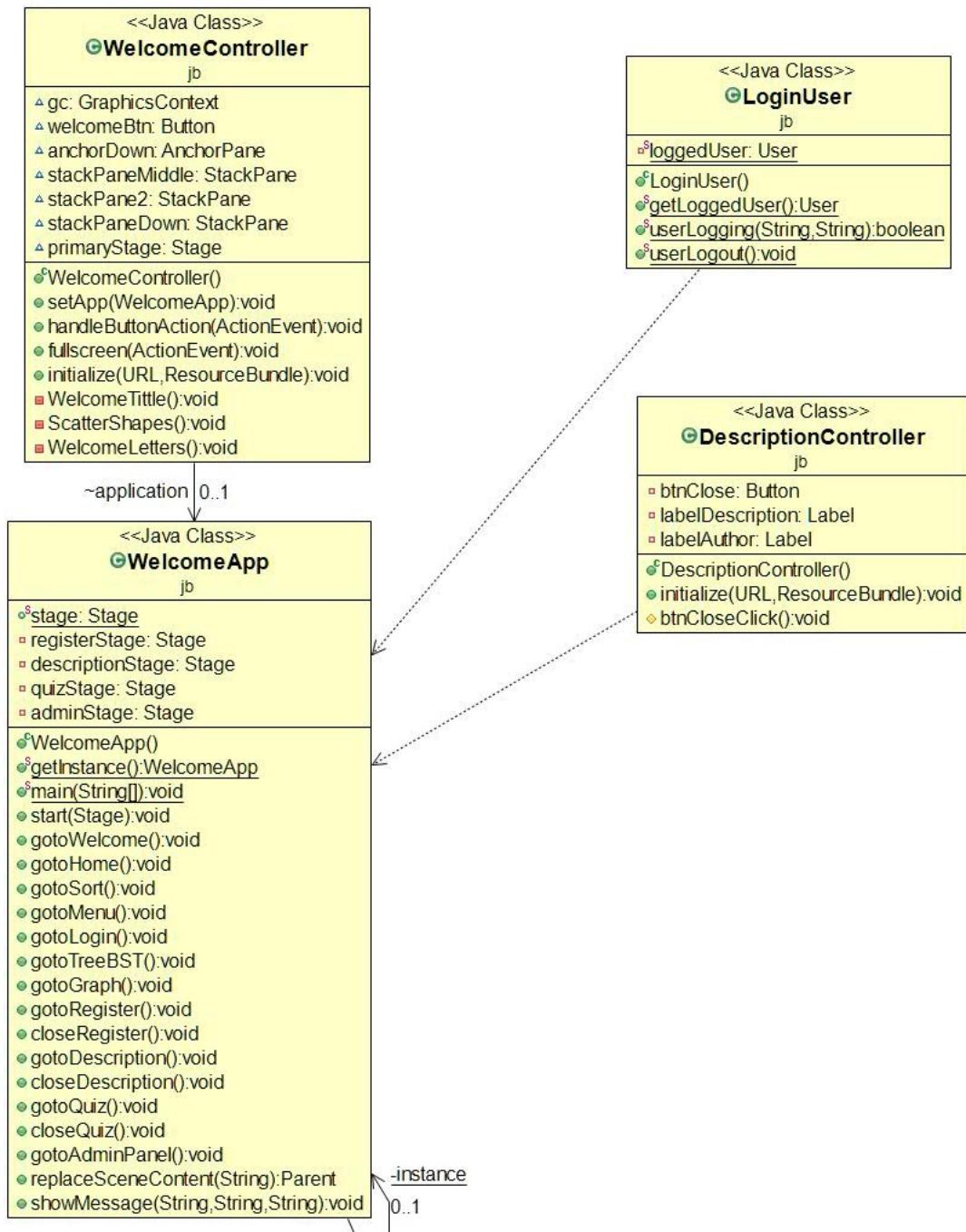


Figure: WelcomeApp.java class

2.3.6 Implementation – class LoginController.java

public class LoginController implements Initializable :

- Class responsible for invoke login/logout user
- method: processLogin() : validates user and login user
- method: processRegister() : open register window with register format
- method: initialize(URL url, ResourceBundle rb) : initialize url, and boundle resources

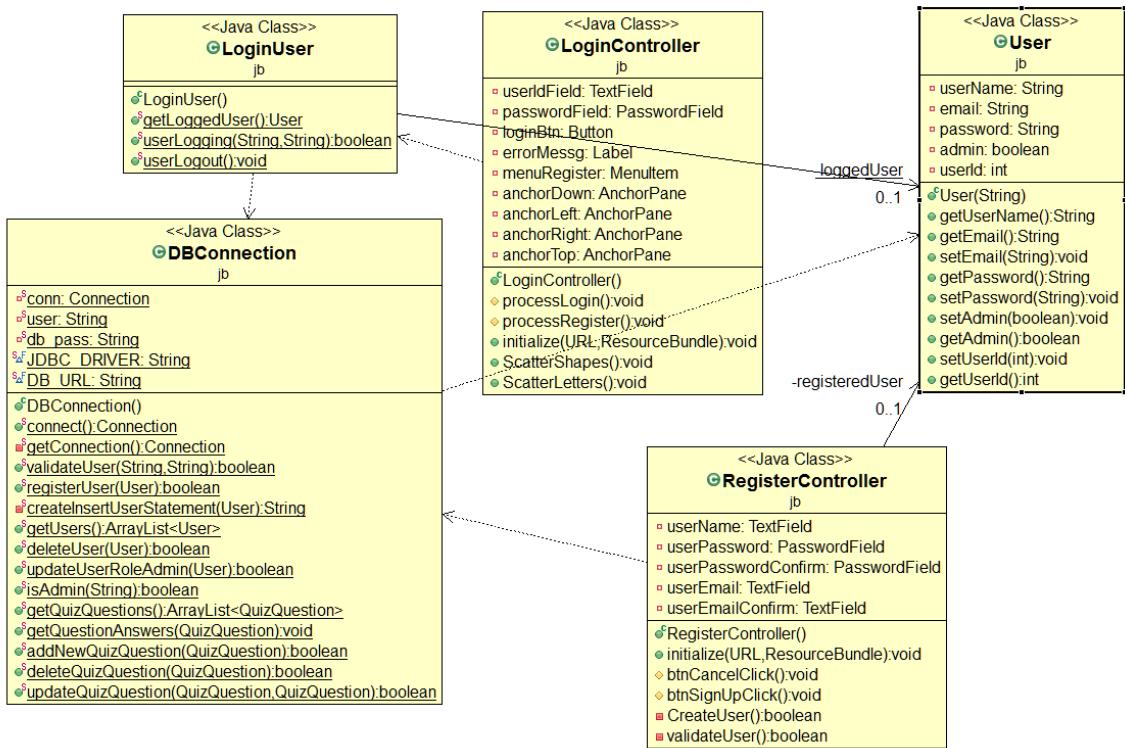


Figure: LoginController.java class

2.3.7 Implementation – class LoginUser.java

public class LoginUser

- Class responsible login/logout user
- method: getLoggedUser() : return logged user which is showed in right corner of each scene

- method: userLogging(String userId, String password) : invoke validate method, and set user as logged if validation passed

- method: userLogout() : logout user

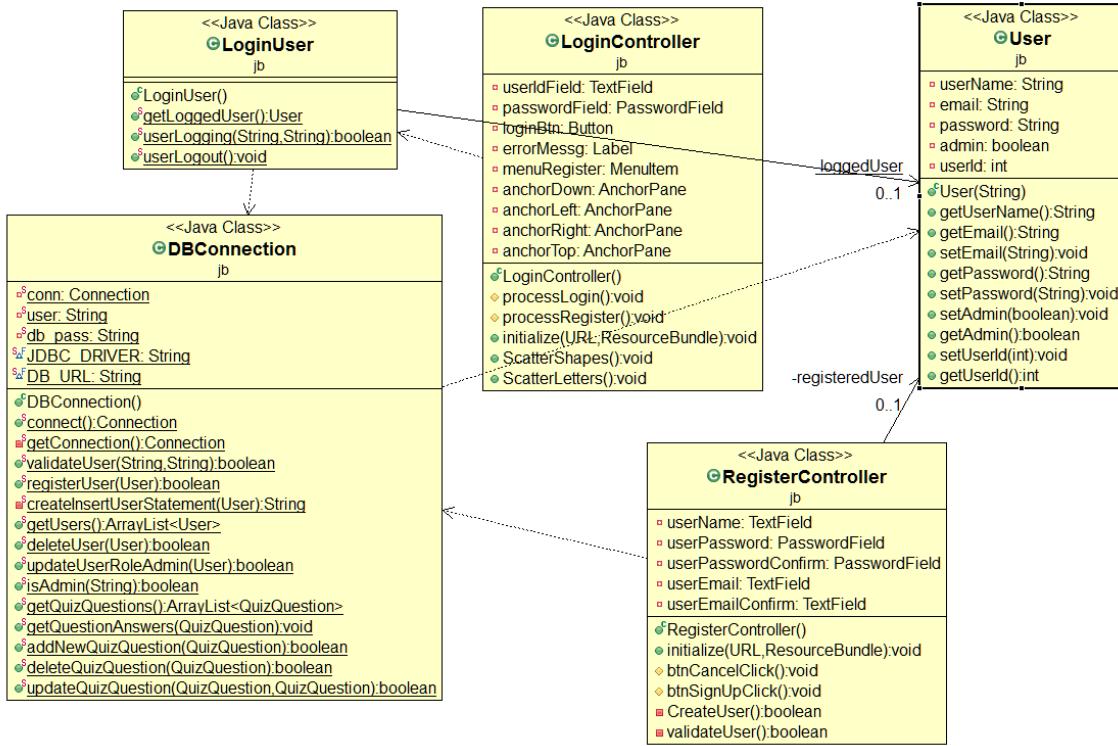


Figure: class LoginUser.java

2.3.8 Implementation – class User.java

```
public class User
```

- Class defines structure of user attributes
- methods: getUserName(), getEmail(), setEmail(String email), getPassword(), setPassword(String password)

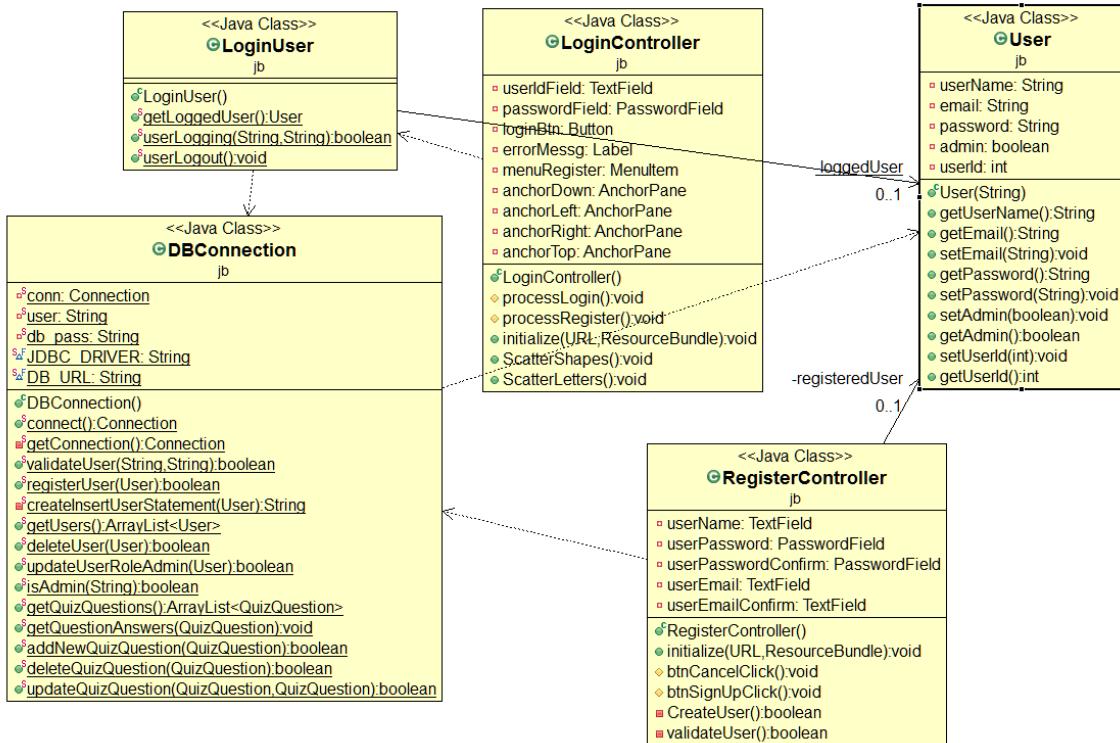


Figure: class User.java

2.3.9 Implementation – class RegisterController.java

```
public class RegisterController implements Initializable :
```

- Class responsible for register user (write/delete user records in database)
- method: `initialize(URL url, ResourceBundle rb)`
- method: `CreateUser()` : create new user, validate user data, insert user data records to database

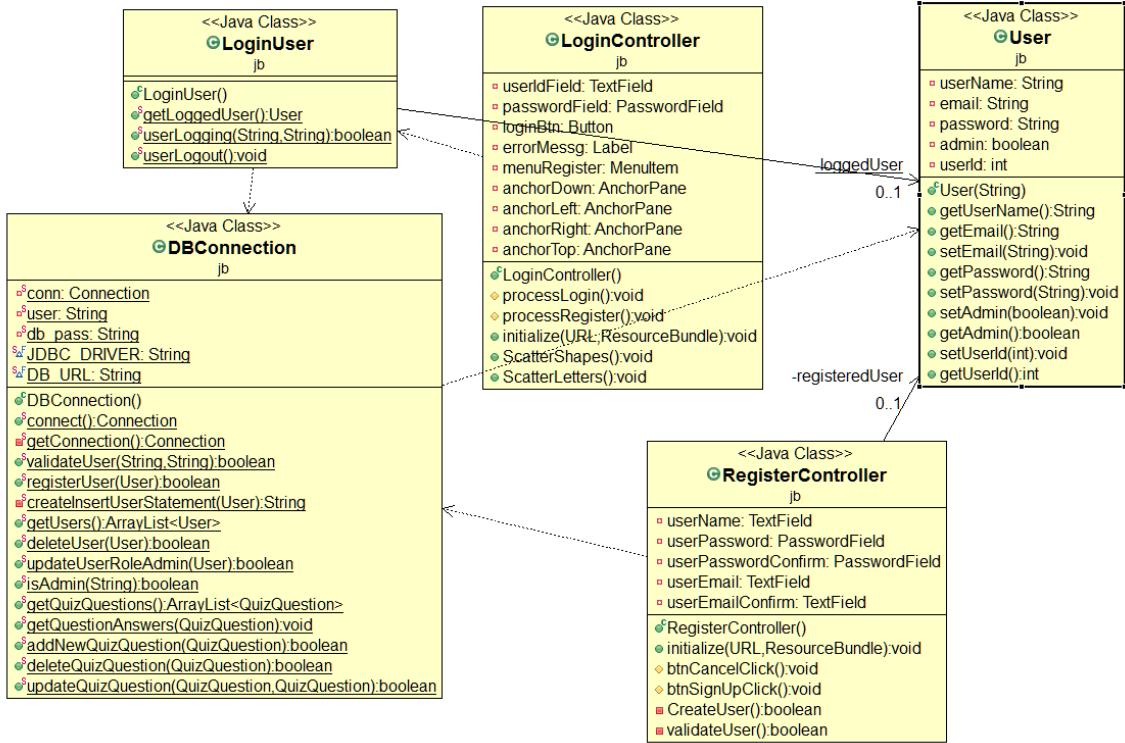


Figure: RegisterController.java

2.3.10 Implementation – class WelcomeController.java

public class WelcomeController implements Initializable :

- Controller class responsible for user invitation,
- method: `handleButtonAction(ActionEvent event)` : opens login window after user clicks enter,
- method: `initialize(URL location, ResourceBundle resources)`,
- method: `WelcomeTittle()` : Animation of text "Welcome in Visual Data" in welcome scene,
- method: `ScatterShapes()` : Animation of shapes in Welcome scene,
- method: `WelcomeLetters()` : Animation of logo in welcome scene,

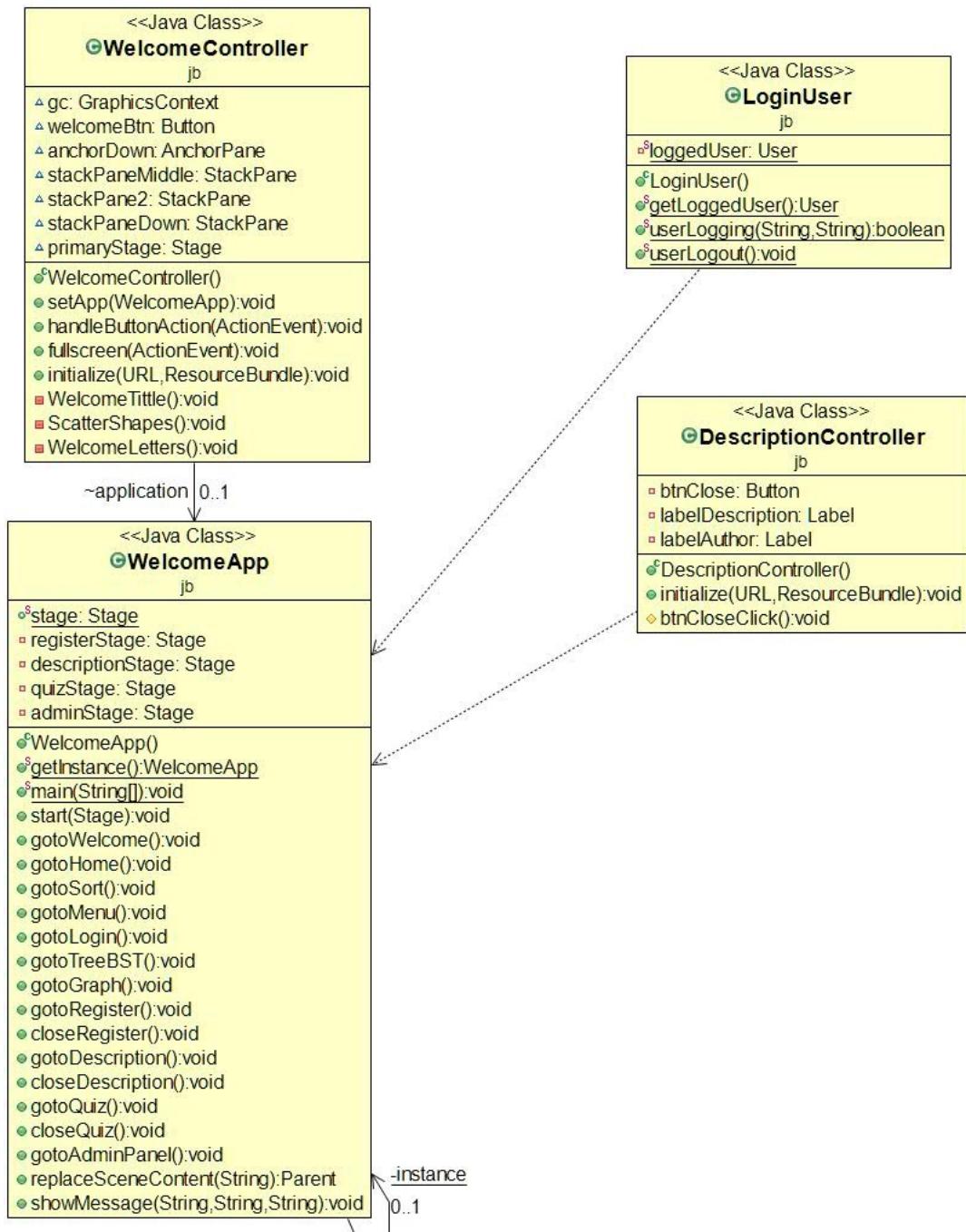


Figure: Class WelcomeController.java

2.3.11 Implementation – class MenuController.java

public class MenuController implements Initializable:

- Controller class responsible for switchover between different functional controllers,
- method: handleQuizBtn(ActionEvent event) : switch to quiz stage
- method: handleDescriptionBtn(ActionEvent event) : switch to stage description
- method: handlesortsBtnBar(ActionEvent event) : switch to SortBars scene
- method: handlesortsBtnPanel(ActionEvent event) : switch to SortPanel scene
- method: handletreeBSTBtn(ActionEvent event) : switch to TreeBST scene
- method: handlegraphsBtn(ActionEvent event) : switch to graph scene
- method: handleLogout() : logout user
- method: handlebubbleSortMenuItem(ActionEvent event): display textual information about bubblesort algorithms
- method: insertionSortMenuItem(ActionEvent event) : display textual information about insertion Sort algorithms
- method: initialize(URL location, ResourceBundle resources) :

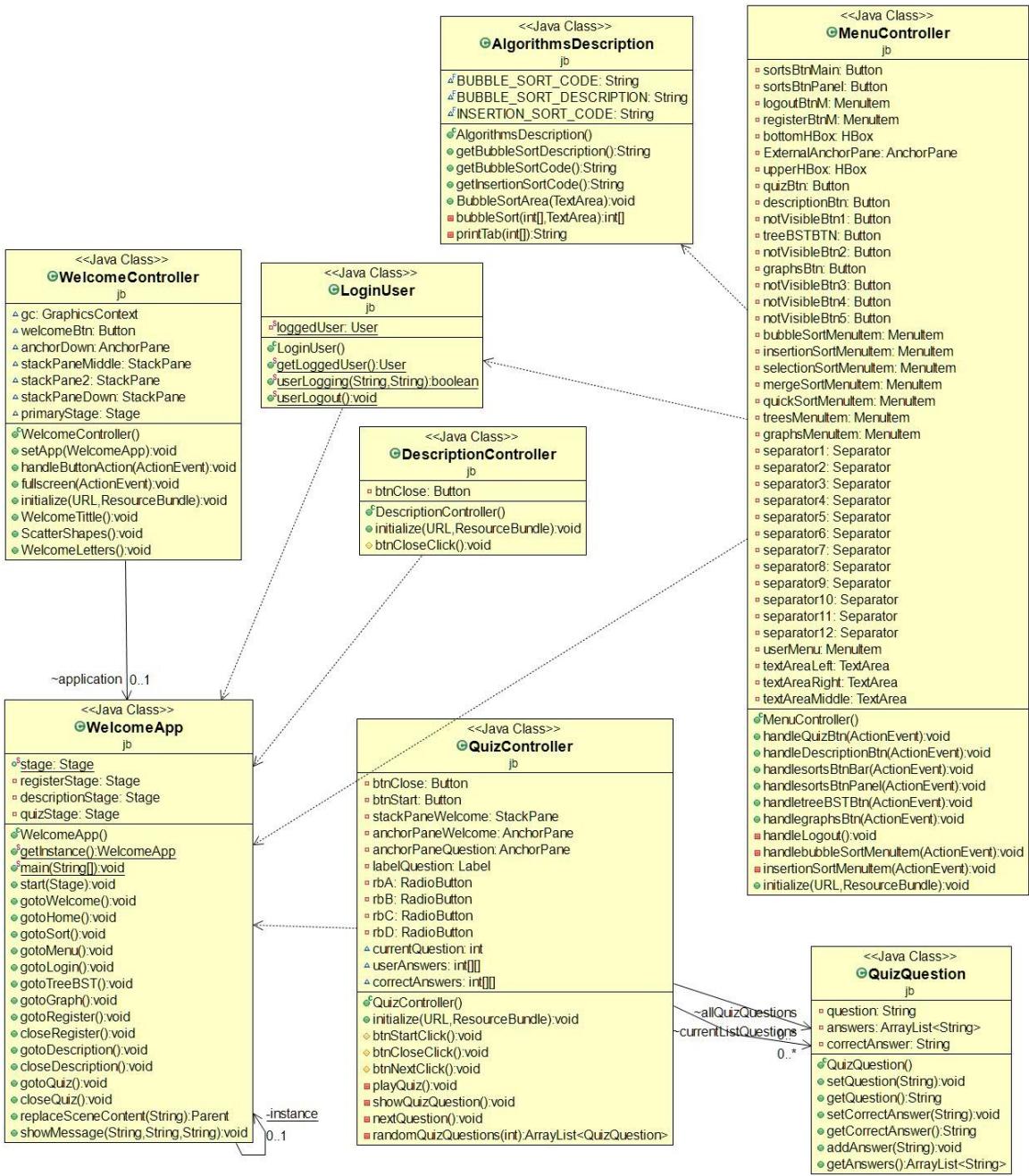


Figure: MenuController.java class

2.3.12 Implementation – class HomeController.java

public class HomeController implements Initializable :

- Controller class which handle sorts on Bars on different sort algorithms. The class manages drawing and animation functions used during sort presentation.
- method: goToHome(ActionEvent event) : switch to Main scene
- method: initialize(URL url, ResourceBundle rb) : initialize graphical controls
- method: handleLogout() : user logout
- method: handleButtonGenerate(ActionEvent event) : generate data to sort
- method: bubbleSorting(ActionEvent event) : invoke bubbleSort sort method, invoke start timer method startTimer for time of sorting
- method: insertionSorting(ActionEvent event) : invoke insertionSorting sort method, invoke start timer method startTimer for time of sorting
- method: selectionSorting(ActionEvent event) : invoke selectionSorting sort method, invoke start timer method startTimer for time of sorting
- method: quickSortingData(ActionEvent event) : invoke quickSortingData sort method, invoke start timer method startTimer for time of sorting
- method: mergeSortingData(ActionEvent event) : invoke mergeSortingData sort method, invoke start timer method startTimer for time of sorting
- method: userSortNodeNumber() : reads input user String SortNumber and converts it to integer
- method: validateUserNumber() : validate input user String SortNumber if it not cross restricted range
- method: generateData() : generate Integer numbers to sort. Generated numbers are kept in the restricted range and are unique. The methods also initialize barchart methods to animate and visualize sorting (drawing barcharts).

The method also activates sort buttons.

- method: AnimationTimer() : method to measure sorting time

- method: drawDataChart(int[] tab) : initialize graphical object barchart with series data in barchart
- method: bubbleSort(int[] a) : implements algorithm bubblesort
- method: insertionSort(int[] a) : implements algorithm insertionSort
- method: selectionSort(int[] a) : implements algorithm selectionSort
- method: quickSort(int[] a) : implements algorithm quickSort
- method: mergeSort(int[] a) : implements algorithm mergeSort

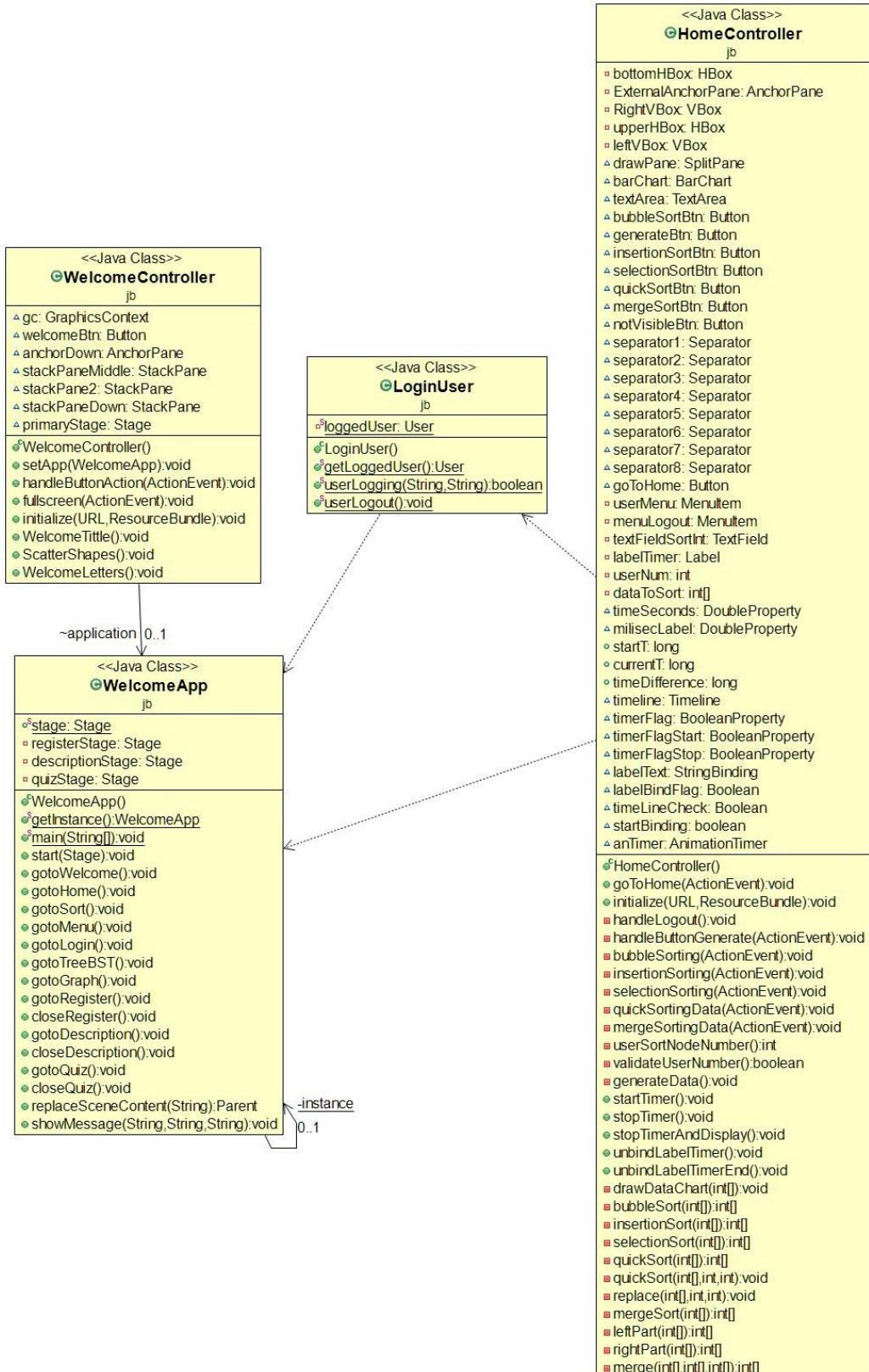


Figure: HomeController class

2.3.13 Implementation – class SortController.java

public class SortController implements Initializable :

- Class Controller which handle sorts on panels on different sort algorithms. The class manages drawing and animation functions used during sort presentation.
- method : initialize(URL url, ResourceBundle rb) : initialize graphical controls
- method: goToHome(ActionEvent event) : switch to Main scene
- method: handleLogout() : user logout
- method: setVisibleSortButtons(boolean visible) : set controls visible
- method: disableSpeedSlider() : disable speed slider control on time of sorting
- method: enableSpeedSlider()() : enable speed slider control when sort is not active
- method: userSortNodeNumber(){ // method return number of data to generate, which user entered in text field
- method: validateUserNumber(){ // method validate if user typed a valid number (range, no letter etc)
- method: handleButtonGenerate(ActionEvent event) : invoke generate method from sort class. Initialize graphical objets to visualization of sorting data . The method also activates sort buttons.
- method: bubbleSorting(ActionEvent event) : invoke bubblesort method from Sort class, and invoke animate method
- method: insertionSorting(ActionEvent event) : invoke insertionSorting method from Sort class, and invoke animate method
- method: selectionSorting(ActionEvent event) : invoke selectionSorting method from Sort class, and invoke animate method
- method: quickSortingData(ActionEvent event) : invoke quickSortingData method from Sort class, and invoke animate method

- method: heapSortingData(ActionEvent event) : invoke heapSortingData method from Sort class, and invoke animate method

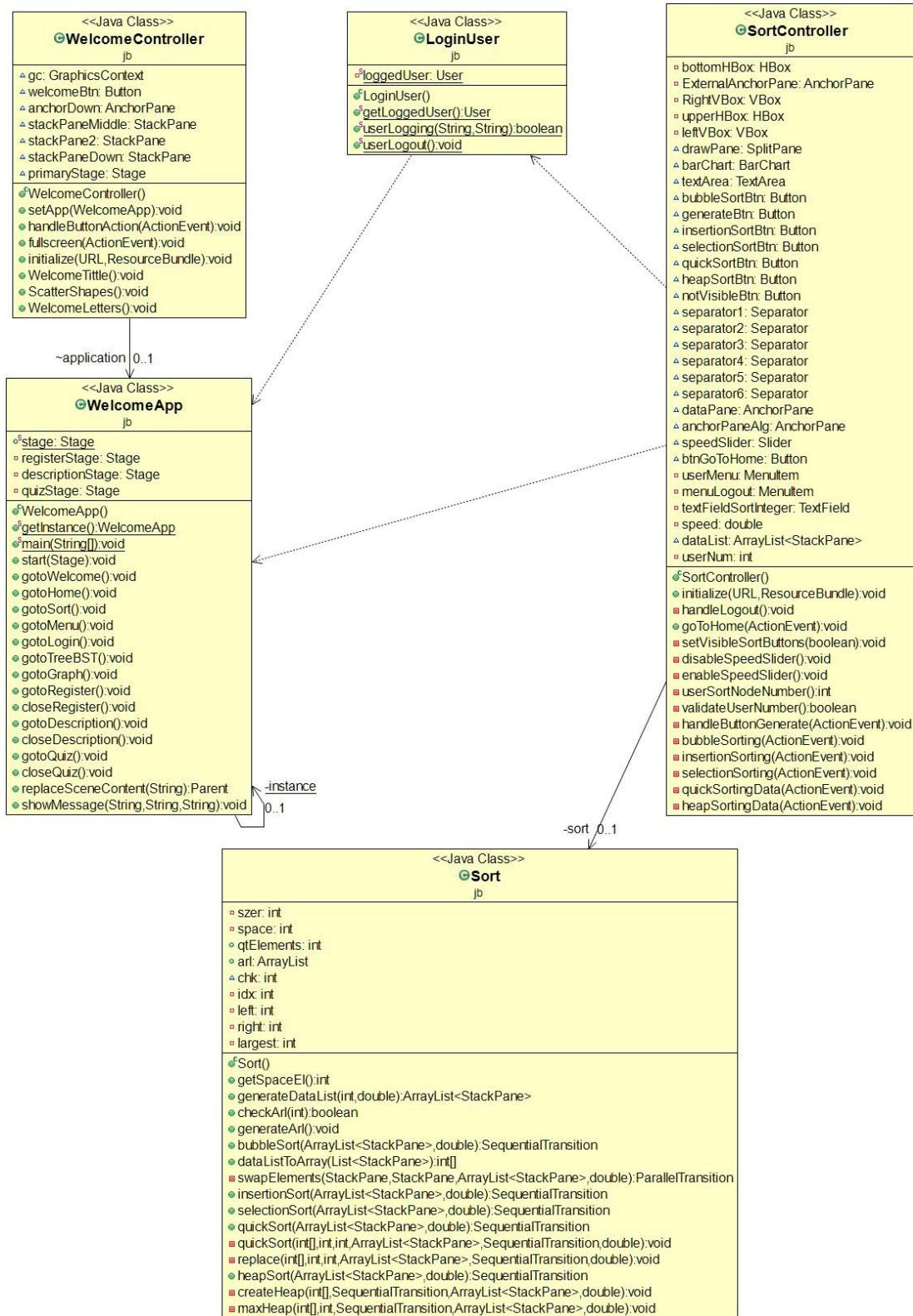


Figure: SortController.java class

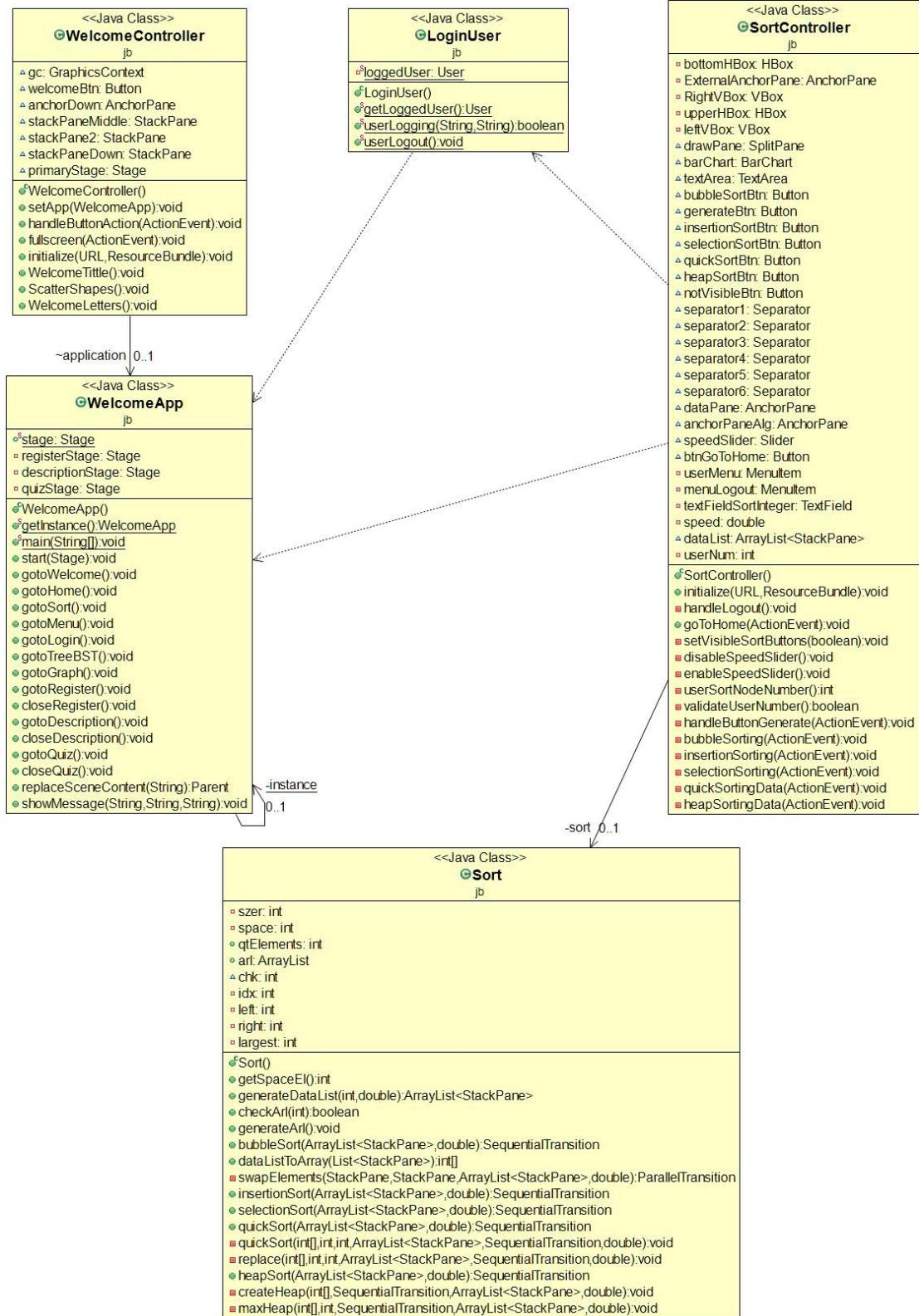
2.3.14 Implementation – class Sort.java

public class Sort :

- Class with sorting methods
 - method: getSpaceEl() : keep space between panels;
 - method: ArrayList<StackPane> generateDataList(int qtElements, double hight) : generate Integer numbers to sort. Generated numbers are kept in the restricted range and are unique. It draws rectangles kept in separate stackpane
 - method: SequentialTransition bubbleSort(ArrayList<StackPane> dataList, double speed): implements bubblesort algorithm , and each swap element is recorded in animation object SequentialTransition, in last play animation of recorded objects.
 - method: SequentialTransition insertionSort(ArrayList<StackPane> dataList, double speed): implements insertionSort algorithm , and each swap element is recorded in animation object SequentialTransition, in last play animation of recorded objects.
 - method: SequentialTransition selectionSort(ArrayList<StackPane> dataList, double speed): implements selectionSort algorithm , and each swap element is recorded in animation object SequentialTransition, in last play animation of recorded objects.
 - method: SequentialTransition quickSort(ArrayList<StackPane> dataList, double speed): implements quickSort algorithm , and each swap element is recorded in animation object SequentialTransition, in last play animation of recorded objects.
 - method: SequentialTransition heapSort(ArrayList<StackPane> dataList, double speed): implements heapSort algorithm , and each swap element is recorded in animation object SequentialTransition, in last play animation of recorded objects.
 - method: dataListToArray(List<StackPane> dataList) : create int array from list of StackPane. The List of StackPanes is replaced to integer array. The sorting

methods use integers from integer array. The animation uses the array list of StackPanels;

- method: swapElements(StackPane sp1, StackPane sp2, ArrayList<StackPane> list, double speed) : swap elements of visualisation. It uses variables share and space calculated in sorting methods



Fugura: Sort.java class

2.3.15 Implementation – class TreeBSTViewController.java

public class TreeBSTViewController implements Initializable .

- Class Controller which implements algorithms for Binary Search Tree, and visualization and animation of Binary Search Tree data structure ,
- method: goToHome(ActionEvent event) : switch to Main scene
- method: initialize(URL url, ResourceBundle rb) : initialize graphical controls
- method: handleLogout() : user logout
- method: getNumberTreeNodes() : returns number of nodes in tree
- method: handleButtonGenerate(ActionEvent event) : invoke method generate draw Binary Search Tree where the number of nodes is entered by user, and invoke draw tree method.
- method: handleButtonInorder(ActionEvent event) : invoke method search in order, and return textual solution of search inorder method.
- method: handleButtonPostOrder(ActionEvent event) : invoke method search post order, and return textual solution of search postorder method.
- method: handleButtonPreOrder(ActionEvent event) : invoke method search post order, and return textual solution of search postorder method.
- method: handleButtonLeaves(ActionEvent event) : invoke method search leaves (printAllRootToLeafPaths()) , and return textual solution of search leaves in tree
- method: handleButtonMaxValue(ActionEvent event) : invoke method maxValue() , and return textual solution of search max value node in tree
- method: handleButtonMinValue(ActionEvent event) : invoke method minValue() , and return textual solution of search min value node in tree
- method : handleButtonTreeHeight(ActionEvent event) : invoke method TreeHeight() , and return textual solution of search height of tree

- method: handleButtonTreeSize(ActionEvent event) : invoke method size() , and return textual solution of search size of tree
- method: handleButtonTreeDepth(ActionEvent event) : invoke method maxDepth() , and return textual solution of search depth of tree
- method: generateTree() : invoke method createTree() if validated entered number of tree is in restricted range
- method: searchInorder() : invoke method drawInOrderGraph()
- method: searchPostOrder() : invoke method: drawPostOrderGraph()
- method: searchPreOrder() : invoke method: drawPreOrderGraph()
- method: searchRootToLeaf() : invoke method: drawRootToLeaf()
- method: searchMaxValueNode() : invoke method: drawMaxValueNode()
- method: searchMinValueNode() : invoke method : searchMinValueNode()
- method: DrawTree(BSTNode n, Color g, SequentialTransition st, StackPane stp) : The recurrence method to draw the tree : first left part is drawn using inorder method, then the method draws lines between nodes if the nodes exist,then right part is drawn using inorder method, then the method draws lines between right nodes if they exist.
- method: centerText(Text text, int x, int y) : keeps text in draw circle
- method: drawInOrderGraph() : draw nodes find in search inorder method
- method: drawPostOrderGraph() : draw nodes find in search postorder method
- method: drawPreOrderGraph() : draw nodes find in search preorder method
- method: drawRootToLeaf() : draw nodes find in search leaves method
- method: drawMaxValueNode() : draw nodes find in search Max value in tree
- method: drawMinValueNode() : draw nodes find in search Min value in tree

2.3.16 Implementation – class BSTNode.java

class BSTNode:

- Class which handle Node in Binary search Tree

2.3.17 Implementation – class BST.java

public class BST:

- Class which handle methods on Binary Search Tree: create tree and searching algorithms

2.3.18 Implementation – class GraphController.java

public class GraphController implements Initializable :

- Class Controller which implements algorithms for Graph, and visualization and animation of Graph data structure ,

The class also implements algorithm Dijkstra's to find shortest paths between nodes in Graph.

- method: initialize(URL url, ResourceBundle rb) : initialize graphical controls
- method: graphControlsSetVisible(boolean visible) : switch visibility of control's buttons
- method: handleLogout() : user logout
- method: goToHome(ActionEvent event) : switch to Main scene
- method: handleButtonGenerate(ActionEvent event) : invoke method createGraph() and DrawGraph(color, indexStart, indexEnd, graphEdgeList)

- method: setStartEndBtnClick(ActionEvent event) : the method takes start node value and end node value from user and validates these values, then draws start and end node in the graph

- method: dijkstraSearchBtnClick(ActionEvent event) : invoke method computeShortestPaths(startIndex) and return textual values of searching method, then draws the shortest paths in the graph between start and end node.

- method: createGraph() : generate of graph from user input value (node numbers). The generation of graph is based on generation of matrix of maximum 100 integer values, but the size depends on user value. The algorithm of the generate matrix first takes user input for comparison, then generate integer value from range 1 to 10, then square the generated value and compares it to user input. If the square value is less than user input the generate matrix method pool once again new value and compares square of it to user input. It is continued until the square pooled value is grater or equal than user input. This method guarantee that for the same input from user every time will be generated different graph. Next the method createGraph() invokes method createRandomGraphNodes(nodeNumbers, matrixSize) which create first nodes for all matrix, then pools from the nodes only the number of nodes provided by user. Next createGraph() method invokes method createEdgesGraphNodes() which creates edges between nodes. The method createEdgesGraphNodes() generates edges random from maximum (n-1) values for each node where n is the number of nodes in graph and choose the edges dynamically.

- method randomMatrixSize(int numberGraphNodes) : The algorithm of the generate matrix first takes user input for comparison, then generate integer value from range 1 to 10, then square the generated value and compares it to user input. If the square value is less than user input the generate matrix method pool once again new value and compares square of it to user input. It is continued until the square pooled value is grater or equal than user input. This method guarantee that for the same input from user every time will be generated different graph.

- method: centerText2(Text text, int xgr, int ygr) : cetner text in drawn nodes.
- method: cleanGraph() : clean drawn graph
- mtehod: userGraphNodesNumber() : input number from user
- method: getPathByNode(GraphNode gp) : return textual values of all nodes on the shortest path between start and end node
- method: DrawGraph(Color clr, int indexStart, int indexEnd, ArrayList<GraphEdge> path) : Draws graph. The drawing of graph is based on drawing nodes according to generated before matrix. The edges are drawn according to generated before method createEdgesGraphNodes() and drawn by method DrawEdge.
- method: DrawEdge(Color color, GraphEdge grEd, int distanceX, int distanceY). The method draws line between nodes, and insert text of edge value to the line
- method : DrawNode(Color color, int graphNodeIndex, int distanceX, int distanceY) : The method draws node for given X and Y value

2.3.19 Implementation – class Graph.java

public class Graph :

- Class implements searching methods on graph and creation of graph
- method: cleanGraphObjects() : clean objects in graph
- method: ArrayList<GraphNode> createRandomGraphNodes(int nodeNumbers, int num) : dynamic generation of nodes in graph
- method: createEdgesGraphNodes() : creates dynamically edges in graph and assign random weights to the edges
- method: correctIndex(int index) : validates user input
- method: computeShortestPaths(int startIndex) : searching shortest paths in graph according to Dijkstra's method
- method: cleanNodesForNewSerach() : clean objects in graph

2.3.20 Implementation – class GraphEdge.java

public class GraphEdge :

- Class keep structure of edge in Graph

2.3.21 Implementation – class GraphNode.java

public class GraphNode :

- Class keep structure of Node in graph

2.3.22 Implementation – class DescriptionController.java

public class DescriptionController implements Initializable :

- Class responsible for presenting description and author details
- method initialize(URL url, ResourceBundle rb) : presents description and author details

2.3.23 Implementation – class QuizController.java

public class QuizController implements Initializable :

- Class handling quiz methods
- method: initialize(URL url, ResourceBundle rb) : initialize graphical elements
- method: playQuiz() : play quiz, read questions and answers from MySQL database, and starts quiz game⁸

- method: showQuizQuestion() : display single question of quiz from MySQL database
- method: randomAnswers() : display possible answers for user in random sequence
- method: reviewQuizQuestion() : displays written answers provided by user and compares with correct value
- method: nextReviewQuestion() : display next review question
- method: nextQuestion() : display next question
- method : summaryQuiz() : summary of the quiz
- method: addUserAnswerToList() : add user answers to the user answers list
- method: ArrayList<QuizQuestion> randomQuizQuestions(int numberQuestions) : generate random questions from the database

2.3.24 Implementation – class QuizQuestion.java

public class QuizQuestion :

- Class keep structure of quiz questions

2.3.25 Implementation – class AlgorithmsDescription.java

public class AlgorithmsDescription :

- Class handles the method for display hints of algorithms

In the class were used wikipedia examples of pseudo-codes⁶ , and descriptions⁶ of used algorithms.

2.4 Testing

Application test consists of two parts.

The first one concerns on the GUI and checks the functionalities of the program Visual Data (VDJB) to ensure it runs and comply to the requirements. The manual tests allowed to assess whether all the planned requirements of application have been implemented.

Verification of the tests was checked according to the following table functionality and test scenarios:

Test Number	Test Description	Passed	Not Passed
Tst1	Functional requirements of the system	V	
Tst2	Access to Visual Data System	V	
Tst3	Start and open program in different screen size mode	V	
Tst4	Log in user	V	
Tst5	Log out user	V	
Tst6	Register user	V	
Tst7	Access to Welcome view	V	
Tst8	Performance of Welcome view	V	
Tst9	Access to Sorts Bars view	V	
Tst10	Performance of Sorts Bars view	V	
Tst11	Algorithms descriptions in Sorts Bars view	V	

Tst12	Enter numbers of elements to sort and generate data in Sorts Bars view	V	
Tst13	Bubble sort method presentation in Sorts Bars view	V	
Tst14	Insertion sort method presentation in Sorts Bars view	V	
Tst15	Selection sort method presentation in Sorts Bars view	V	
Tst16	Merge sort method prestenattion in Sorts Bras view	V	
Tst17	Quick sort method presentation in Sorts Bars view	V	
Tst18	Bubble sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst19	Insertion sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst20	Selection sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst21	Merge sort method prestenattion in with maximum number of elements Sorts Bras view	V	
Tst22	Quick sort method presentation in with maximum number of elements Sorts Bars view	V	
Tst23	Bubble sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst24	Insertion sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst25	Selection sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst26	Merge sort method prestenattion in with minimum number of elements Sorts Bras view	V	
Tst27	Quick sort method presentation in with minimum number of elements Sorts Bars view	V	
Tst28	Reliability of the system when all buttons are pressed when program is running	V	
Tst29	Time of sorting visibility for each sort method in Sorts Bars view	V	
Tst30	Switch back to Menu option from Sorts Bars view	V	
Tst31	Access to Sorts Panels view	V	
Tst32	Performance of Sorts Panels view	V	
Tst33	Enter numbers of elements to sort and generate data in Sorts Panels view	V	
Tst34	Bubble sort method presentation in Sorts Panels view	V	
Tst35	Insertion sort method presentation in Sorts Panels view	V	
Tst36	Selection sort method presentation in Sorts Panels view	V	
Tst37	Heap sort method prestenattion in Sorts Panels view	V	
Tst38	Quick sort method presentation in Sorts Panels view	V	

Tst39	Slide speed change of animation panels	V	
Tst40	Low speed Bubble sort method presentation in Sorts Panels view	V	
Tst41	Low speed Insertion sort method presentation in Sorts Panels view	V	
Tst42	Low speed Selection sort method presentation in Sorts Panels view	V	
Tst43	Low speed Heap sort method prestenattion in Sorts Panels view	V	
Tst44	Low Quick sort method presentation in Sorts Panels view	V	
Tst45	High speed Bubble sort method presentation in Sorts Panels view	V	
Tst46	High speed Insertion sort method presentation in Sorts Panels view	V	
Tst47	High speed Selection sort method presentation in Sorts Panels view	V	
Tst48	High speed Heap sort method prestenattion in Sorts Panels view	V	
Tst49	High Quick sort method presentation in Sorts Panels view	V	
Tst50	Low speed Bubble sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst51	Low speed Insertion sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst52	Low speed Selection sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst53	Low speed Heap sort method prestenattion in with maximum number of elements Sorts Panels view	V	
Tst54	Low speed Quick sort method presentation in with maximum number of elements Sorts Panels view	V	
Tst55	High speed Bubble sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst56	High speed Insertion sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst57	High speed Selection sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst58	High speed Heap sort method prestenattion in with minimum number of elements Sorts Panels view	V	
Tst59	High speed Quick sort method presentation in with minimum number of elements Sorts Panels view	V	

Tst60	Low speed Bubble sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst61	Low speed Insertion sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst62	Selection sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst63	Low speed Heap sort method prestenattion in with maximum number of elements Sorts Panels view	V	
Tst64	Low speed Quick sort method presentation in with maximum number of elements Sorts Panels view	V	
Tst65	Low speed Bubble sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst66	Low speed Insertion sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst67	Low speed Selection sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst68	Low speed Heap sort method prestenattion in with minimum number of elements Sorts Panels view	V	
Tst69	Low speed Quick sort method presentation in with minimum number of elements Sorts Panels view	V	
Tst70	Switch back to Menu option from Sorts Panels view	V	
Tst71	Access to Trees view	V	
Tst72	Performance of Trees view	V	
Tst73	Enter numbers of elements to generate BST Tree in Trees view	V	
Tst74	Inorder traversal option with path presentation in Trees view	V	
Tst75	Postorder traversal option with path presentation in Trees view	V	
Tst76	Preorder traversal option with path presentation in Trees view	V	
Tst77	Leaves selected in Trees view	V	
Tst78	Find minimum value in Trees view	V	
Tst79	Find maximum value in Trees view	V	
Tst80	Height of tree presentation in Trees view	V	
Tst81	Size of tree presentation in Trees view	V	
Tst82	Depth of tree presentation in Trees viw	V	
Tst83	Switch back to Menu option from Trees view	V	
Tst84	Access to user management for logged user	V	
Tst85	Delete user when user is logged	V	
Tst86	Set Admin role for user	V	

Tst87	Access to quiz questions management	V	
Tst88	Add new quiz question	V	
Tst89	Update quiz question and its answers	V	
Tst90	Delete quiz question	V	
Tst91	Access to Graphs view	V	
Tst92	Performance of Graphs view	V	
Tst93	Enter numbers of elements to generate Graph in Graphs view	V	
Tst94	Performance of search methods in Graphs view	V	

Table: Manual tests Visual Data System

Second part of the testing included automatic testing supported by JUnit - framework for Java programming language. JUnit 4 is integrated in NetBeans IDE 8.1. The NetBeans IDE 8.1 was used for creating and testing application. JUnit unit testing framework enables quickly and easily create test suites.

It increases the quality of code and software reliability. I choosed a few critical classes to build unit tests in this method, especially all the classes responsible for the Searching Sorts and Binary Search Tree algorithms. Each test class corresponds to the application class. The test class consists of test Initializers and Finalizers, and methods for unit cases. Initialization method runs in the tests class before each test case, test finalizer method is running after each test case in the test class. The Junit tests are very helpful when there is need to initialize some variables before run test or to clean up any data when finish.

In JUnit4 each test method should have an @Test annotation if we want it to take part in the running tests.^{1,2}

JUnit tests examples:

Class BSTTest

Initiate BST the established values: 8, 3, 10, 1, 6, 14, 4, 7, 13 to build tree:

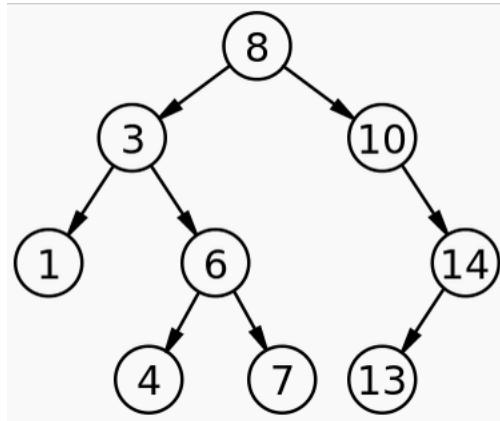


Figure: BST tree

Methods:

setUp() - to init BST before unit case

testGetRoot() - method return a root - should return 8

testIsEmpty() - method check if tree is empty - should return false

testSearch() - method search data in BST, for value 6 - should return true, for value 5 - false;

testSize() - method return numbers of elements in BST, should return 9

testMaxDepth() - method return depth of BST, the longest path from root to leaf, should return 4

testMinValue() - should return 1

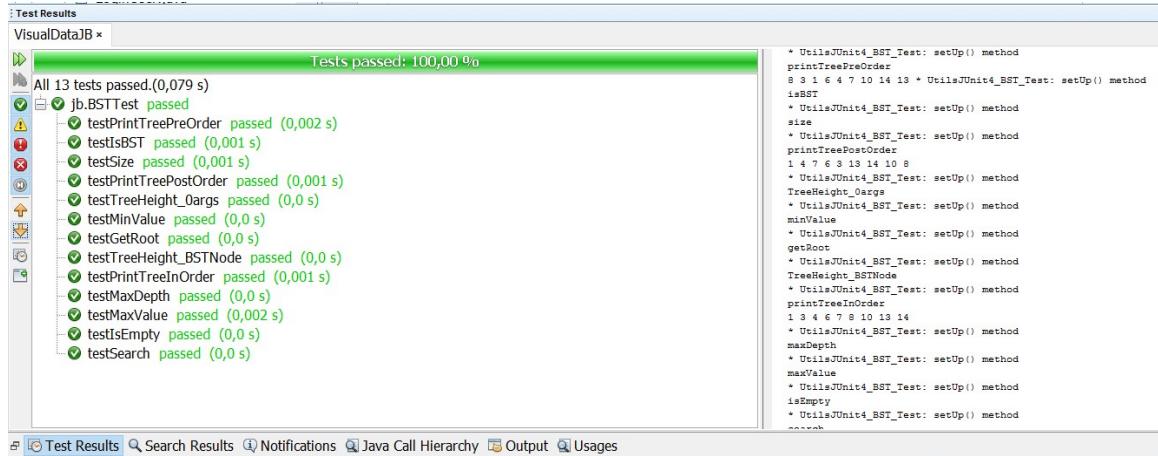
testMaxValue() - should return 14

testPrintTreeInOrder() - method traverse BST in In-Order method, should return path: 1, 3, 4, 6, 7, 8, 10, 13, 14

`testPrintTreePreOrder()` - method traverse BST in Pre-Order method, should return path: 8, 3, 1, 6, 4, 7, 10, 14, 13

`testPrintTreePostOrder()` - method traverse BST in Post- Order method, should return path: 1, 4, 7, 6, 3, 13, 14, 10, 8

Execution of the test shown in the figure:



Class SortTest

In this class I build an init array which fixed values, it helps test my sort method.

Methods:

`setUp()` - to init an array of 12 values before unit case:

```
int[] tab = {8, 3, 10, 1, 6, 14, 4, 7, 13, 2, 19, 5};
```

This array is used to build data list to sort in test methods.

`testBubbleSort()` – sorts data list using algorithm Bubble sort, returns sorted list

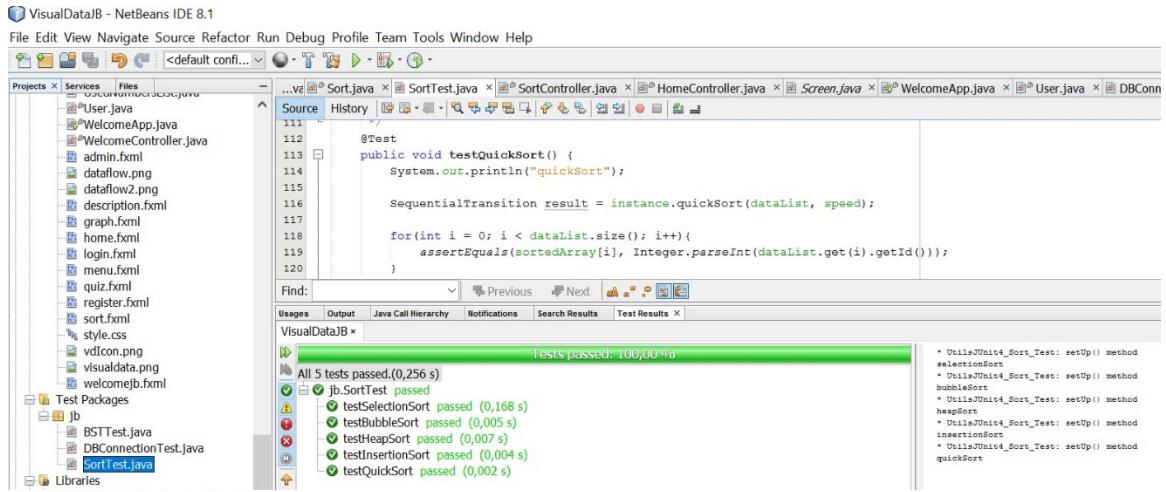
`testInsertionSort()` – sorts data list using algorithm Insertion sort, returns sorted list

`testSelectionSort()` – sorts data list using algorithm Selection sort, returns sorted list

`testQuickSort()` – sorts data list using algorithm Quick sort, returns sorted list

`testHeapSort()` - sorts data list using algorithm Heap sort, returns sorted list

Execution of this test shown in the figure:



Class DBConnectionTest

In this class I test a connection with data base and methods which interact with database.

This class doesn't init any variables in `setUp()` method.

Tested method:

`testConnect()` - method checks if connection is correctly created

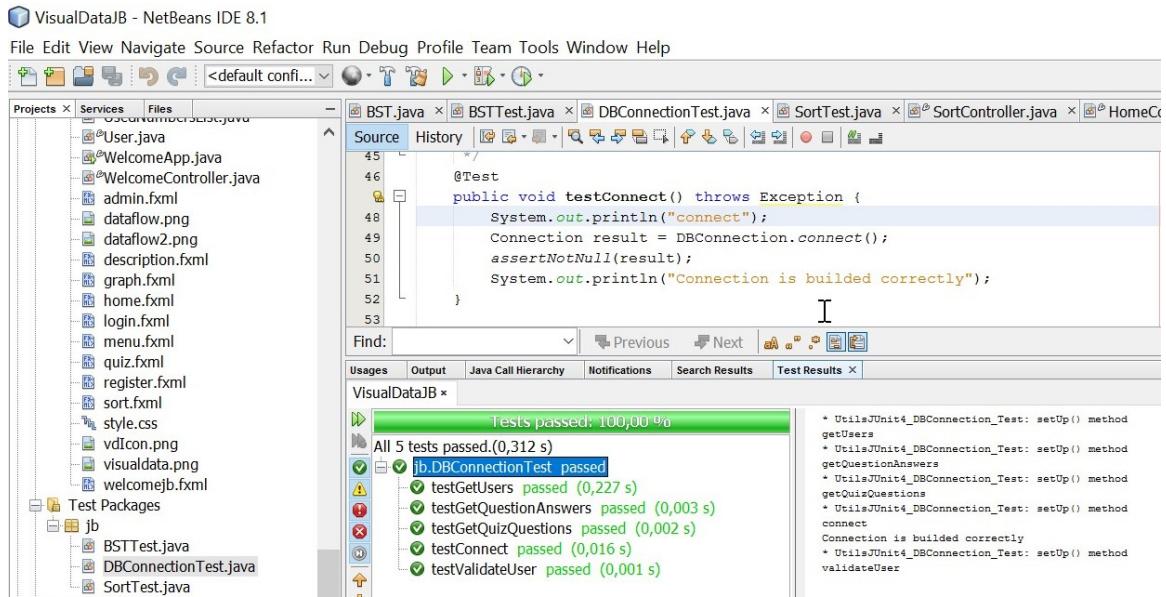
`testValidateUser()` – method validate if user is registered in database

`testGetUsers()` – method returns users from database, count of users should be
 > 0

`testGetQuizQuestions()` - method returns quiz questions from database, count of questions should be > 0

`testGetQuestionAnswers()` – method tests if question has exactly 4 answers

Execution of this test shown in the figure:



2.5 Graphical User Interface (GUI) Layout

Program was developed in Rich JavaFX language environment. .

2.6 Customer testing

The tool was tested by undergraduate student NCIRL Computer Science.

2.7 Evaluation

The system was evaluated by a group of undergraduate students of Computer Science at the National College of Ireland.

Printed an anonymous questionnaire was distributed among students. Students were asked to save the anonymous feedback form.

The form listed the evaluation questions about satisfaction with the program, defects in the program, expected improvements in the system. System was rated positively. Nice design of the system was marked as a major advantage.

Second place went to animation algorithms and visualization of tree and graph.

Another feature of a well-evaluated this possibility to change the speed of the animation and display execution time of algorithms.

Students suggested making a few improvements of the system.

To this list we can include:

- Implement methods to remove elements from the data structures,
- Implement traversing algorithm showing parts of the running code,
- Implementing additional data structures: queues, linked lists, AVL tree, stack.

2.8 SWOT

Strengths

Practical and universal subject of the program is based on the fundamental algorithms and data structures Modular construction project the possibility of further expansion use graphical capabilities Java FX

Weaknesses

Limited use of graphics capabilities due to the time limit of the project were inadequate compared to the experience gained in graphics solutions limited experience in automated testing of GUI

Threads

The access to MySQL may be limited.

Opportunities

The functionality of the application may be expanded with new features.

3 Conclusions

This paper presents an animated tutoring system that can be used by students to expand their knowledge of algorithms and data structures. The system is based on the animation of sorting algorithms and graphical visualization of trees and graphs. The system shows the visualization methods of searching trees and graph.

The system also provides the user with basic information about the algorithms and data structures. Tutorial has been enhance in the quiz with questions and answers in the field of algorithms and data structures.

The rating obtained in an anonymous evaluation highlights the tutorial as a useful tool in learning and teaching.

This tool is a potential help for students to understand the theoretical aspects of algorithms and data structures.

The results from the students' survey helped to define further steps in development of the tool. The development will include following improvements of the system:

- Implement methods to remove elements from the data structures,
- Implement traversing algorithm showing parts of the running code,
- Implementing additional data structures: queues, linked lists, AVL tree, stack.

4 References

1. <http://www.tutorialspoint.com/junit/> [Accessed December 2016].
2. <https://netbeans.org/kb/docs/java/junit-intro.html> [Accessed December 2016].
3. <http://www.cgpundit.com/animation-in-education/> [Accessed December 2016].
4. http://www.educationworld.com/a_curr/profdev/profdev094.shtml [Accessed December 2016].
5. <https://www.teachervision.com/reading-comprehension/skill-builder/48791.html> [Accessed December 2016].
6. <https://www.wikipedia.org/> [Accessed December 2016]
7. <http://www.oracle.com/introduction-to-fxml> [Accessed December 2016]
8. <http://engineeringinterviewquestions.com/data-structures-and-algorithms-multiple-choice-questions-and-answers/> [Accessed December 2016]
9. https://en.wikipedia.org/wiki/The_Art_of_Computer_Programming [Accessed December 2016]
10. Data Structures & Algorithms in Java by Robert Lafore , Sams, 1988
11. Data Structures and Algorithms in Java, Michael T. Goodrich, Department of Computer Science University of California, Irvine, John Wiley & Sons, Inc., 2004.

5 Appendix

Attach all your partial submissions as appendices.

5.1 Project Proposal

Project Proposal

INTERACTIVE SOFTWARE APPLICATION FOR DATA STRUCTURE VISUALIZATION AND ALGORITHM ANIMATION

JACEK BYZDRA, 15030491, x15030491@student.ncirl.ie

Degree Programme Name: Higher Diploma in Science in Computing

Specialisation : Software Development

Date : 02-10-2016

5.1.1 Objectives

The project presents an interactive software application designed to visualization of commonly used data structures. One of the application feature is animation of used algorithms. This environment can be used as very useful educational tool in Computer Science.

The tool implements interactive feature that allows user to change methods, and data to trace, and compare efficiency of different algorithms. The pedagogical aim of this tool is to facilitate the student's grasp programming of common data structures and algorithms. The algorithms are key in majority of computer programs. It is well known that learning data structures and algorithms brings students a lot of consternation.

The tool is dedicated to the study of the main data structures by use interactive interface to guide in Java programming language. The application can be used in training courses presenting commonly used data structures and algorithms.

It is planned the software will illustrate in a graphical, visual, interactive, and animation way the following well-known data structures :

- linear data structures (arrays, stacks, queues, lists),
- tree-like data structures (binary search trees, AVL trees, hash trees),
- graphs

Additionally, the animations incorporated in this tool will illustrate sorting algorithms. It is planned also of executing defined algorithms and visualize the state of the data structures used by the algorithms.

Below algorithms will be implemented in this software application :

- Sorting (Bubble sort, Selection sort, Insertion Sort, Quick Sort, Merge Sort)
- Searching (Binary Search Tree, Linear Search, Dijkstra's Shortest Path)
- Deleting
- Inserting

The main learning possibility of this tool is the interactive execution of data structures and algorithms. The system will display data structures graphically

providing the facility to allow users to perform the basic operations on the data structure generated. It will be possible also to change a set of animations that show how data structures are used . An animation-based approach will be developed to provide a step-by-step illustration of how various traversal techniques are performed in the context of used algorithms. The tool will present data structure implementation both in a static and dynamic way.

5.1.2 Background

In 1990, at the Helsinki University of Technology was developed system TRAKLA, software supporting the data structures and algorithms. It was the first system to implement computerized algorithm simulation exercises. The student simulated the working of algorithms on a conceptual level, originally performed manually with pen an paper. In final state of the simulation the system just checked the answers which were submitted to the system as email messages.

In 1994-95 Jun Yang and Cliff Shaffer designed a system SWAN, a data structure visualization system developed as part of Virginia Tech's NSF Educational Infrastructure project. The system was based on C/C++ language to support visualization of graphs, including arrays, lists, trees and general graphs.

In 1998 at the University of Siegen was developed system ANIMAL, a compact, and easy to use animation tool to generate animations.

The motivation for developing ANIMAL came from a lack of satisfying tools for animations that are both easy to generate and edit and helpful in lectures. The available object or animation effect types were not coded directly. The animations could be automatically generated using either the scripting language ANIMALSCRIPT or the ANIMALGENERATOR API. The program might appear somewhat limited in the animation effects as well as the primitive graphic types offered. ANIMAL was implemented completely in Java language.

In 2001 Becker and Beacham found that it is often difficult to let student understand a knowledge of the creation and operation of data structures by using traditional method. To fill up the gap they developed a graphical, animated system aiming to provide a step-by-step walkthrough operations on B-Tree data structure.

In 2002 in Department of Computer Science, Duke University, was developed system visual and interactive tools to be used by instructors to teach computer science and by students to learn concepts in a visual and hands-on manner. JAWAA was a simple command language for creating and movement of primitive objects (circles, lines, text, rectangles) and displaying them with a Web browser. Commands are stored in a script file that is retrieved and run by the JAWAA applet when the applet's Web page is accessed through the Web.

In 2009 del Vado Virseda developed an intelligent tutoring system which guide the interactive learning of data structures. The possibility to use an animated tutoring system has triggered several efforts from the computer science community to develop web-based animated systems. But in most web sites the data structures and algorithms are presented typically in static techniques, such as slides or textual information.

The well known sites which present data structures in interactive way are presented below :

- <https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>
- <http://visualgo.net/>
- <http://www.algoanim.ide.sk/>
- <https://www.toptal.com/developers/sorting-algorithms>

These Web sites are dedicated on presentation of Data Structures and Algorithms based on Java Scripts, and HTML interfaces. The changes in graphics elements, option used in interactive window, used data base for tests, and rich GUI interface is mostly here limited.

5.1.3 Technical Approach

The approach is to design interactive application showing use of pseudocode in visualization/animation of flows in data structure.

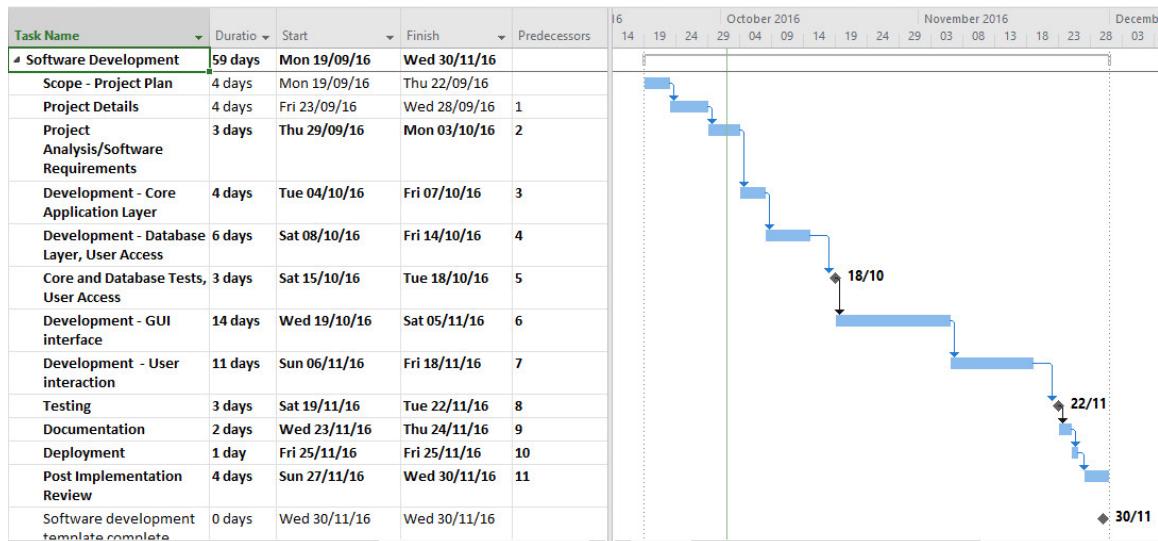
The planned software application will use the following techniques :

- Visualization of widely used data structures : array, stack, queue, tree, graph, with use bar charts, graphs, and other graphic elements.
- Animation window which presents animation of chosen methods used on data structures : insert, delete, search, sort. The animation shown in stepwise or in a continuous way,
- Interactive application with multiple choice : different algorithms, data structures, capacity of data , time, comparison of algorithms,
- Changes in color and illumination of graphic elements,
- Presentation of methods used in the algorithms: Graphical explanation of flows in data structures,
- Explanation window with a brief description of data structures and algorithms,
- Pseudocode window which presents code used in algorithms,
- The help window with information about used controls (after mouse flyover or button press),
- Window login/logout for user and administrator. User can use application after login,
- Execution of several data structures/algorithms and several sequences of operations at the same time making use of a multi-windows and multi-frame system,

The interactive methods, where student may trace, step by step the operation of algorithms should be very useful both for teaching and learning. The planned system which may assist in training of commonly used data structures and algorithms.

The software application will be implemented in Graphical User Interface based on AWT, SWING, JavaFX Java language making it platform independent and portable.

5.1.4 Project Plan



5.1.5 Technical Details

The proposed software application will be written in Java AWT, Java Swing, JavaFX and Java applet. For demonstration purpose the part of code will be used with communication with MySQL database.

5.1.6 Evaluation

In order to obtain a detailed evaluation of the usage of the system, I have proposed several tests related to the behavior, implementation and application of the main data structures offered by the tool.

The evaluation will be carried out also by computer science undergraduate students in the School of Computing.

JACEK BYZDRA

October 2nd, 2016

5.2 Project Plan

Task Name
▪ Software Scope
Project
Project Analysis
Requirements
Development Application
Development Layer, Core and User API
Development interface
Development interface
Testing Documentation
Deployment
Post Implementation Review
Software Manual

National College of Ireland

Technical Report

INTERACTIVE SOFTWARE APPLICATION FOR DATA STRUCTURE VISUALIZATION AND ALGORITHM ANIMATION

5.3 Requirements Specification

Jacek Byzdra

Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
10/09/2016	1	Create	ABCD	X	X

Distribution List

Name	Title	Version
Anu Sahni	Lecturer	
Glen Ward	Lecturer	

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Table of Contents

1	Introduction	228
1.1	Purpose	228
1.2	Project Scope	228
1.3	Definitions, Acronyms, and Abbreviations	229
2	User Requirements Definition	229
3	Requirements Specification	230
3.1	Functional requirements	230
3.1.1	Use Case Diagram	28
3.1.2	Requirement 1 <name of requirement in a few words>	29
3.1.3	Requirement 2 <name of requirement in a few words> Błąd! Nie zdefiniowano zakładki.	
3.2	Non-Functional Requirements	321
3.2.1	Performance/Response time requirement	321
3.2.2	Availability requirement	321
3.2.3	Recover requirement	321
3.2.4	Robustness requirement	321
3.2.5	Security requirement	321
3.2.6	Reliability requirement	322
3.2.7	Maintainability requirement	322
3.2.8	Reusability requirement	322
4	GUI	Błąd! Nie zdefiniowano zakładki.
5	System Architecture	Błąd! Nie zdefiniowano zakładki.
6	System evolution	Błąd! Nie zdefiniowano zakładki.
7	References	

2 Introduction

2.1 Purpose

The purpose of this document is to set out the requirements for “INTERACTIVE SOFTWARE APPLICATION FOR DATA STRUCTURE VISUALIZATION AND ALGORITHM ANIMATION” (Visual Data) software. It will illustrate the purpose and complete declaration for the development of system and also explain system interactions with other external applications.

2.2 Project Scope

The project presents an interactive software application designed to visualization of commonly used data structures. One of the application feature is animation of used algorithms. The tool implements interactive feature that allows user to change methods, and data to trace, and compare efficiency of different algorithms, and show data structures.

System should illustrate in a graphical, visual, interactive, and animation way the following data structures:

- Binary Search Tree,
- Graphs

The tool should illustrate following sorting algorithms in a graphical, visual, interactive, and animation way:

- Bubble Sort,
- InsertionSort,
- Selection Sort,
- Merge Sort,
- Quick Sort,
- Heap Sort,

System should perform below operations on Binary Search Tree:

- In order search,
- Post order search,
- Pre order search,
- Search leaves in Binary Search Tree,
- Search minimum value,
- Search maximum value,
- Search height of the Binary Search Tree,
- Search size of the Binary Search Tree,
- Search depth of the Binary Search Tree,

System should perform quiz with multiple questions , answers in the field of algorithms and data structures. The quiz should presents correct and not correct answers provided by user.

2.3 Definitions, Acronyms, and Abbreviations

BST – Binary Search Tree

Dijkstra's – Algorithm found by Edsger Wybe Dijkstra

3 User Requirements Definition

The user requirements presents requirements of the system from user perspective. It includes services, knowledge required to operate the application. The application presents visualisation and animation of algorithms and data structures, designed and coded in Java language.

The level of presented information requires from user basic knowledge of data structures and algorithms and basic knowledge of Java language. To read instructions and notes presented in the the application user should handle English language on medium level.

4 Requirements Specification

This section contains all of the functional and quality requirements of the system. The section gives a detailed description of the system and all its features. All GUI's and interfaces should require minimum training.

4.1 Functional requirements

These system requirements are categorised by its necessity for the app to operate.

- Cat1: Mandatory – The app will not serve its primary function in the absence of this requirement;
- Cat2: Ideal – Adds value to the app but does not impact on its primary function or operation;
- Cat3: Novel – Increases the attraction of the app but does not provide any functionality.

The list of the required functional requirements of the system is presented below:

No	Description of Functional requirement	When it is required ?	Dependencies	Category
FR1	Visual Animation of Bubble Sort Algorithm (Sorts Bars View). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Bubble Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR2	Visual Animation of Insertion Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1

FR3	Visual Animation of Selection Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR4	Visual Animation of Merge Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Merge Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR5	Visual Animation of Quick Sort Algorithm (Sorts Bars view). Show sorting in ascending order	When the program runs in Sorts Bars Window, and user press Quick Sort button	System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window	1
FR6	Show time of sorting operation in Sorts Bars Window	When the program runs in Sorts Bars Window, and user press the sort button for each algorithm. System displays time of the opeartion when sortinmg is finished	System is running, User is logged in, Sorts Bars window is opened, User invoked sorting of any algorithm before	1
FR7	Switch to Menu window from Sorts Bars window after home button is pressed	When the program runs and Sorts Bars window is opened. System should switch to Menu window independently of running sorting algorithms	User is logged in, Sorts Bars window is opened, System is running and Sorts Bars Window is opend	1
FR8	Maximize Window when maximize button is pressed	When the program runs , and is not in a register mode	System is running. Register window is not opened	2
FR9	Minimize Window when minimize button is pressed	When the program runs , and is not in a register mode	System is running. User is logged in, Register window is not opened	2

	Generate Data for sorting bars when the Generate Data button is pressed	When the program runs and Sorts Bars window is open, user provided correct integer number to sort before, and user pressed generate data button	System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label	
FR10	Generate Data for sorting bars when the Generate Data button is pressed	When the program runs and Sorts Bars window is open, user provided correct integer number to sort before, and user pressed generate data button	System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label	1
FR11	Display information "Incorrect sort number"	When the program runs and Sorts Bars window is open, user provided incorrect integer number to sort before, and user pressed generate data button	System is running, user is logged in and Sorts Bars Window is open, user provided incorrect integer number to sort label	1
FR12	Display Buttons Generate Data, Home, and Integer Sort Label in Sorts Bars window	When the program runs in Sorts Bars Window	System is running, user is logged in and the Sorts Bars window is opened	1
FR13	Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window	When the program runs in Sorts Bars window, and user entered and generated correct integer number to sort	When the program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort	1
FR14	Display information about logged user in Sorts Bars window	When the program runs, user is logged in, and Sorts Bars window is open	When the program runs, user is logged in, and Sorts Bars window is open	1
FR15	Log user out after Logout button is pressed	When the program runs, user is logged in, and Sorts Bars window is open, and user pressed Logout button	When the program runs, user is logged in, and Sorts Bars window is open, and user pressed Logout button	1
FR16	Visual Animation of Bubble Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Bubble Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1

	Visual Animation of Insertion Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	
FR17	Visual Animation of Selection Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Insertion Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR18	Visual Animation of Heap Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Merge Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR19	Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Quick Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR20	Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order	When the program runs in Sorts Panels Window, and user press Quick Sort button	System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window	1
FR21	Switch to Menu window from Sorts Panel window after home button is pressed	When the program runs and Sorts Panels window is opened. System should switch to Menu window independently of running sorting algorithms	User is logged in, Sorts Panels window is opened, System is running and Sorts Panels Window is open	1
FR22	Generate Integer Data for sorting bars when the Generate Data button is pressed	When the program runs and Sorts Panels window is open, user provided correct integer number to sort before, and user pressed generate data button	System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label	1

	Display information "Incorrect sort number"	When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button	System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label	
FR23	Display Buttons Generate Data, Home, and Integer Sort Label in Sorts Panels window	When the program runs in Sorts Panels Window	System is running., user is logged in and the Sorts Panels window is opened	1
FR24	Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Panels window	When the program runs in Sorts Panels window, and user entered and generated correct integer number to sort	When the program runs, user is logged in in Sorts Panels window, and user entered and generated correct integer number to sort	1
FR25	Display information about logged user in Sorts Panels window	When the program runs , user is logged in, and Sorts Panels window is open	When the program runs , user is logged in, and Sorts Panels window is open	1
FR26	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Sort Bars window is open, and user pressed Logout button	When the program runs , user is logged in, and Sorts Bars window is open, and user pressed Logout button	1
FR27	Visual Creation of Binary Search Tree from generated integer data	When the program runs in Binary Search Tree window, user provided before corrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided correct number to BST search label	1
FR28	Visual Animation of Inorder Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Inorder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR29				

	Visual Animation of Post Order Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed PostOrder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	
FR30	Visual Animation of Pre Order Search Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed PreOrder button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR31	Visual Animation of Search Leaves Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Leaves button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR32	Visual Animation of Search Max value of nodes Algorithm (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Max button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR33	Visual Animation of Search Max value of node in BST(Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Max button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR34	Visual Animation of Search of Min value of node in BST (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Min button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR35	Visual Animation of Search of Height of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Height button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	
FR36	Display textual information of Height of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Height button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1

FR37	Textual display of size of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Size button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR38	Textual display of depth of Binary Search Tree (Binary Search Tree View).	When the program runs in Binary Search Tree window, user created Binary Search Tree before and pressed Depth button	System is running, user is logged in, Binary Search Tree window is opened , user created before BST tree	1
FR39	Generate number of nodes in the Binary Search Tree when the generate data button is pressed	When the program runs in Binary Search Tree window, user provided before corrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided correct number to BST search label	1
FR40	Display information "Enter correct number of nodes"	When the program runs in Binary Search Tree window, user provided before incorrected number of nodes to generate, and pressed Generate Data button	System is running, user is logged in, Binary Search Tree window is opened , user provided incorrect number to BST search label	1
FR41	Display buttons Generate Data, Home, Inorder, PostOrder, PreOrder, leaves, Max, Min, Height, Size, Depth, Label number of nodes in Binary Search Tree window	When the program runs in Binary Search Tree window,	System is running, user is logged in, Binary Search Tree window is opened	1
FR42	Display information about logged user in Binary Search Window	When the program runs , user is logged in, and Binary Search Tree window is opend	System is running, user is logged in, Binary Search Tree window is opened	1
FR43	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Binary Search Tree window is opend, and	When the program runs , user is logged in, and Binary Search window is opend, and user pressed	1

		user pressed Logout button	Logout button	
FR44	Switch to Menu window from Binary search Tree window after home button is pressed	When the program runs and Binary Search Tree window is opened. System should switch to Menu window after home button is pressed independently of running searchich methods in BST window	User is logged in, Binary Search Tree window is opened, System is running	1
FR45	Visual Creation of Graph from generated integer data	When the program runs in Graphs window, user provided before corrected number of nodes to generate, and pressed Generate Graph button	System is running, user is logged in, Graphs window is opened , user provided correct number to graph search label	1
FR46	Switch to Menu window from Graphs window after home button is pressed	When the program runs and Graphs Panels window is opened. System should switch to Menu window independently of running graph searching algorithms	User is logged in, Graphs window is opened, System is running	1
FR47	Generate Integer Data for creating graph when the generate button is pressed	When the program runs and Graphs Panels window is open, user provided correct integer number to generate graph before, and user pressed generate graph button	System is running, User is logged in, and Graphs Window is open, user provided correct integer number to graph label	1
FR48	Display information "Incorrect graph size If you want generate graph please enter correct graph sizer"	When the program runs and Graphs window is open, user provided incorrect integer number	System is running, user is logged in and Graphs Window is open, user provided incorrect	1

		to create graph before, and user pressed generate graph button	integer number to create graph label	
FR49	Display Buttons Generate Graph Home, and Integer Graph Label in Graphs window	When the program runs in Graphs Window	System is running., user is logged in and the Graphs window is opened	1
FR50	Display Button:Start Node Label, End Node Label, Set start/end button, Dijkstra's Shortest Paths button in Graphs window	When the program runs in Graphs window, and user entered and generated correct integer number to create graph	When the program runs, user is logged in in Graphs window, and user entered and generated correct integer number to create graph	1
FR51	Display information about logged user in Graphs window	When the program runs , user is logged in, and Graphs window is open	When the program runs , user is logged in, and Graphs window is open	1
FR52	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Graphs window is open, and user pressed Logout button	When the program runs , user is logged in, and Sorts Bars window is open, and user pressed Logout button	
FR53	System visualize the start, end node, and the shortest path in different color. Displays textual information about number of nodes in Graph, number of edges in Graph, Displays textual information about sum of weights of paths from start Node to every Node. Next system displays textual information abot the shortest path from start to end node, information about weights of the shortest path, and information about which Nodes are in the shortest path.	When Dijkstra's Shortest Paths button is pressed System visualize the start, end node, and the shortest path in different color, displays textual information about number of nodes in Graph, number of edges in Graph, Displays textual information about sum of weights of paths from start Node to every Node. Next system displays textual information abot the shortest path from start to end node,	When the program runs, user is logged in, and Graphs window is opened, and user provided correct number of nodes to create graph, and press button generate graph	1

		information about weights of the shortest path, and information about which Nodes are in the shortest path.		
FR54	Visualize start , end node in Graphs window	When graph is created and correct start, end node value is provided to label graph and set start/end button is pressed system displays start , end node in differnet color	When the program runs, user is logged in, and Graphs window is opened, and user provided correct number of nodes to create graph, and press button set start/end node in graph	1
FR55	Display information: please enter correct index for start node and for end node and click Set button, when the provided value of start end node was incorrect	When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button	When the program runs, user is logged in, and Graphs window is opened, and user provided incorrect number of nodes to create graph, and press buttonet start/end node in graph	1
FR56	Display Buttons: Quiz, Description, Sorts Bars, Sorts Panels, Trees, Graphs, Hints: Bubble Sort, Hints, logged userl in Menu window	When the program runs in Menu Window	System is running., user is logged in and the Menu window is opened	1
FR57	Display Button:Hints: Insertion Sort, Hints Selection Sort, Hints merge Sort, Hints Quick Sort, Hints Trees, Hints Graph in Menu window	When the program runs in Menu Window and Hints button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR58	Display information about logged user in Menu window	When the program runs , user is logged in, and Menu window is opend	When the program runs , user is logged in, andMenu window is opend	1
FR59	Logged user out after Logout button is pressed	When the program runs , user is logged in, and Menu window is opend,	When the program runs , user is logged in, and Menu window is opend,	1

		and user pressed Logout button	and user pressed Logout button	
FR60	Switch to Sorts Bars window from Menu Window after Sorts Bars button is pressed	When the program runs and Menu window is opened. System should switch to Sorts Bars window after Sorts Bars button is pressed	User is logged in,Menu window is opened, System is running	1
FR61	Switch to Sorts Panels window from Menu Window after Sorts Panels button is pressed	When the program runs and Menu window is opened. System should switch to Sorts Panels window after Sorts Panels button is pressed	User is logged in,Menu window is opened, System is running	1
FR62	Switch to Tree window from Menu Window after Tree button is pressed	When the program runs and Menu window is opened. System should switch to Tree window after Tree button is pressed	User is logged in,Menu window is opened, System is running	1
FR63	Switch to Graphs window from Menu Window after Graph button is pressed	When the program runs and Menu window is opened. System should switch to Graphs window after Graph button is pressed	User is logged in,Menu window is opened, System is running	1
FR64	Switch to Quiz window from Menu Window after Quiz button is pressed	When the program runs and Menu window is opened. System should switch to Quiz window after Quiz button is pressed	User is logged in,Menu window is opened, System is running	1
FR65	Display information about system when Description button is pressed	When the program runs and Menu window is opened System should display information about the system	User is logged in,Menu window is opened, System is running	1

FR66	Display Hints information about Bubble Sort algorithm when Hints Bubble Sort button is pressed	When the program runs in Menu Window and Hints Bubble Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR67	Display Hints information about Insertion Sort algorithm when Hints Insertion Sort button is pressed	When the program runs in Menu Window and Hints Bubble Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR68	Display Hints information about Selection Sort algorithm when Hints Selection Sort button is pressed	When the program runs in Menu Window and Hints Selection Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR69	Display Hints information about Merge Sort algorithm when Hints Merge Sort button is pressed	When the program runs in Menu Window and Hints Merge Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR70	Display Hints information about Quick Sort algorithm when Hints Quick Sort button is pressed	When the program runs in Menu Window and Hints Quick Sort button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR71	Display Hints information about BST Trees data structure when Hints Trees button is pressed	When the program runs in Menu Window and Hints Trees button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR72	Display Hints information about Graphs data structure when Graphs button is pressed	When the program runs in Menu Window and Hints Graphs button is pressed	When the program runs, user is logged in, and Hints button is pressed	1
FR73	Display Welcome Window with animation and visualization of system logo, and display Enter button when the application is started	When user starts the application.	System is not started yet.	1
FR74	Display information about author of the program in top of each window , except Register window	When system is running the information about author of the program should be displayed in each window except register window	System is running, register Window is not opened	2

FR75	Switch to Login Window from Welcome Window when Enter button is pressed in Welcome Window	When the program runs in Welcome Window and Enter button is pressed	Program runs in Welcome Window	1
FR76	Display Login Window with login button, user Name label, Password label , and Register button in Login Window	When the program runs in Welcome Window and Enter button is pressed	Program runs in Welcome Window, Enter button is pressed	1
FR77	Switch to Register Window when register button is pressed in Login Window	When the program runs in Login Window and Register button is pressed	Program runs in Login Window, Register button is pressed	1
FR78	Display fields: User Name, Password, Confirm Password, E-mail, Confirm E-mail, and button Cancel, and Sign Up in Register Window	When the program runs in Login Window and Register button is pressed	Program runs in Login Window, Register button is pressed	1
FR79	Display information: User name is incorrect. Please enter correct user name for register when the user name was not provided in registration form in Registration Window and Sign Up button was pressed	When the program runs in Register Window and user not provided user name and pressed Sign Up button	Program runs in Register Window	1
FR80	Display information: Password is empty when password was not provided to the system, after user name filled user name field and pressed Sign Up button in Register Window	Display information: Password is empty when password was not provided to the system, after user name filled user name field and pressed Sign Up button in Register Window	Program runs in Register Window	1
FR81	Display information: Incorrect Password confirmation when Confirm Password was not provided correctly to registration form, after user name, and password was filled and Sign up button was pressed in Register Window	Display information: Incorrect Password confirmation when Confirm Password was not provided correctly to registration form, after user name, and password was filled and Sign up	Program runs in Register Window	1

		button was pressed in Register Window		
FR82	Display information: Email address is empty when Email address was not provided to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Registration Window	Display information: Email address is empty when Email address was not provided to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Registration Window	Program runs in Register Window	1
FR83	Display information: Incorrect Email address when Email address was not provided correctly to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Register Window	Display information: Incorrect Email address when Email address was not provided correctly to registration form, after user name, and password, and confirmation password was filled and Sign up button was pressed in Register Window	Program runs in Register Window	1
FR84	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Program runs in Register Window	1

		Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, or provided incorrectly, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Display information: We need confirmation your Email address when confirmation of Email address was not provided to registration form, or provided incorrectly, after user name, password, confirmation password, and email was filled and Sign up button was pressed in Register Window	Program runs in Register Window	1
FR85		Display information: You are registered in this application Please log in now when all user provided correctly all required fields in registration form and pressed Sign Up button in registration window	Display information: You are registered in this application Please log in now when all user provided correctly all required fields in registration form and pressed Sign Up button in registration window	Program runs in Register Window	1
FR86		Switch to Login Window from registration form when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Switch to Login Window from registration form when user successfully provided all required information in registration form and pressed Sign up button in Registration Window	Program runs in Register Window	1
FR87		Switch to Login Window from registration form when user pressed cancel button in Registration Window	Switch to Login Window from registration form when user pressed cancel button in Registration Window	Program runs in Register Window	1
FR88		Register user in MySQL database visualdatajb when user successfully provided all	Register user in MySQL database visualdatajb when user successfully provided all	Program runs in Register Window, user filled correctly all required	1
FR89					

	required information in registration form and pressed Sign up button in Registration Window	provided all required information in registration form and pressed Sign up button in Registration Window	fields in registration form in registry Window and pressed Sign Up button	
FR90	Display information: User/Password combination is not valid. Are you new ? Please use register option, when username or password was not provided or provided not correctly to login fields in LoginWindow	Display information: User/Password combination is not valid. Are you new ? Please use register option, when username or password was not provided or provided not correctly to login fields in LoginWindow	Program runs in Login Window	1

4.1.1 Use Case Diagrams

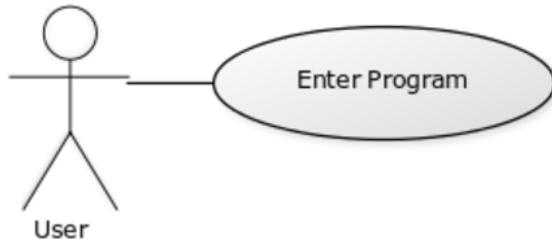


Figure 1 Use Case Diagram FR73

4.1.2 Requirement 73: "Welcome invitation - FR73"

4.1.2.1 Description & Priority

When the user starts to use the system it will display a welcome screen, with the enter button. The welcome screen animates logo and title of the welcome window.

4.1.2.2 Use Case

Scope

The scope of this use case is to invite the user.

Description

This use case describes the process of invitation.

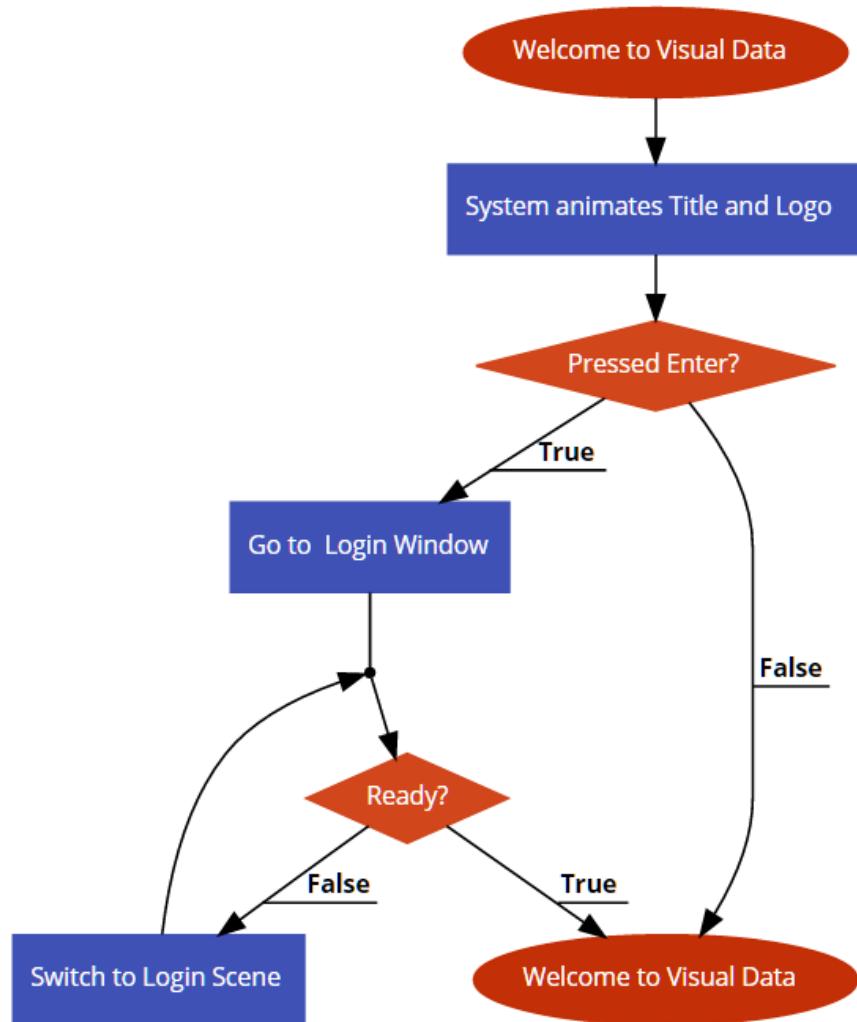


Figure 2 Flow Diagram FR73

Termination Flow Description

Precondition

The user has opened system.

Activation

This use case starts when a user opens system.

Main flow

A1. The system starts the application.

A2. The system animates title and logo

Alternate flow

B1: The system starts the application.

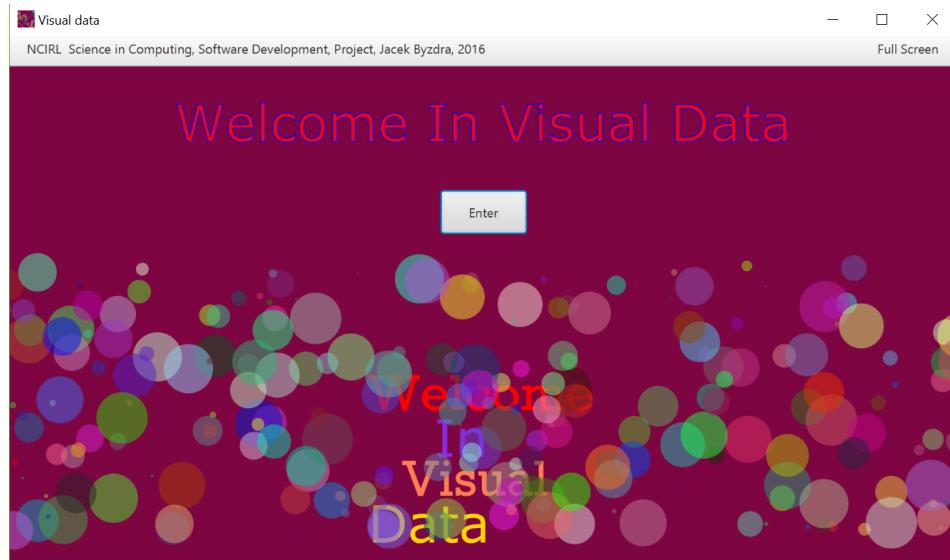
Termination

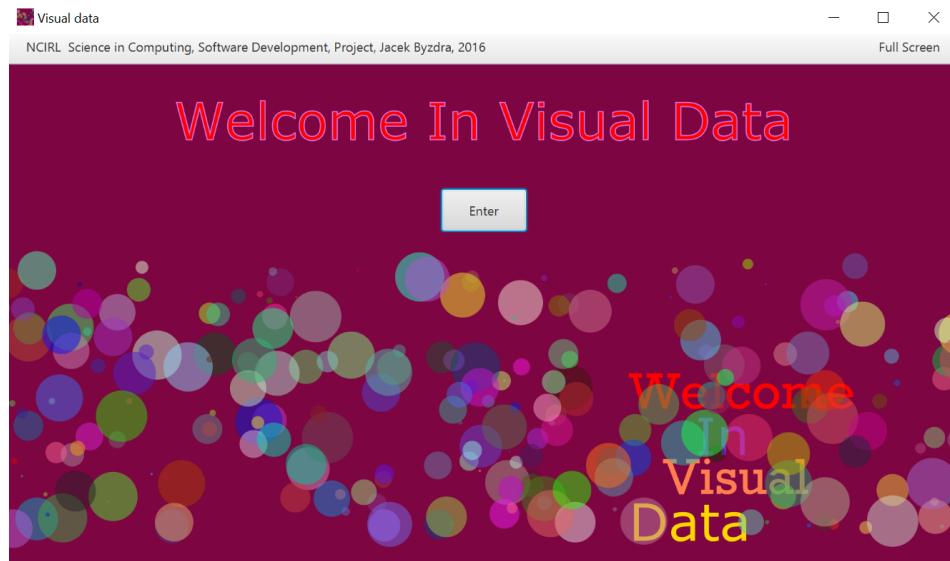
The user press Enter button

Post condition

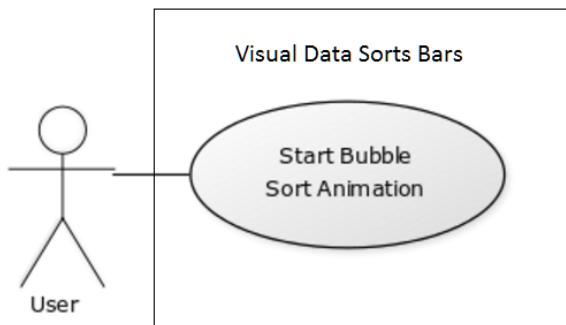
The system animates title and logo

Post condition Mock





4.1.3 Requirement 1:"FR1"



4.1.3.1 Description & Priority

Visual Animation of Bubble Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Bubble Sort button. System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

4.1.3.2 Use Case

Scope

The scope of this use case is to start sort Bubble Sort in Sorts Bars view.

Description

This use case describes the process of sorting algorithm Bubble Sort in Sorts Bars window

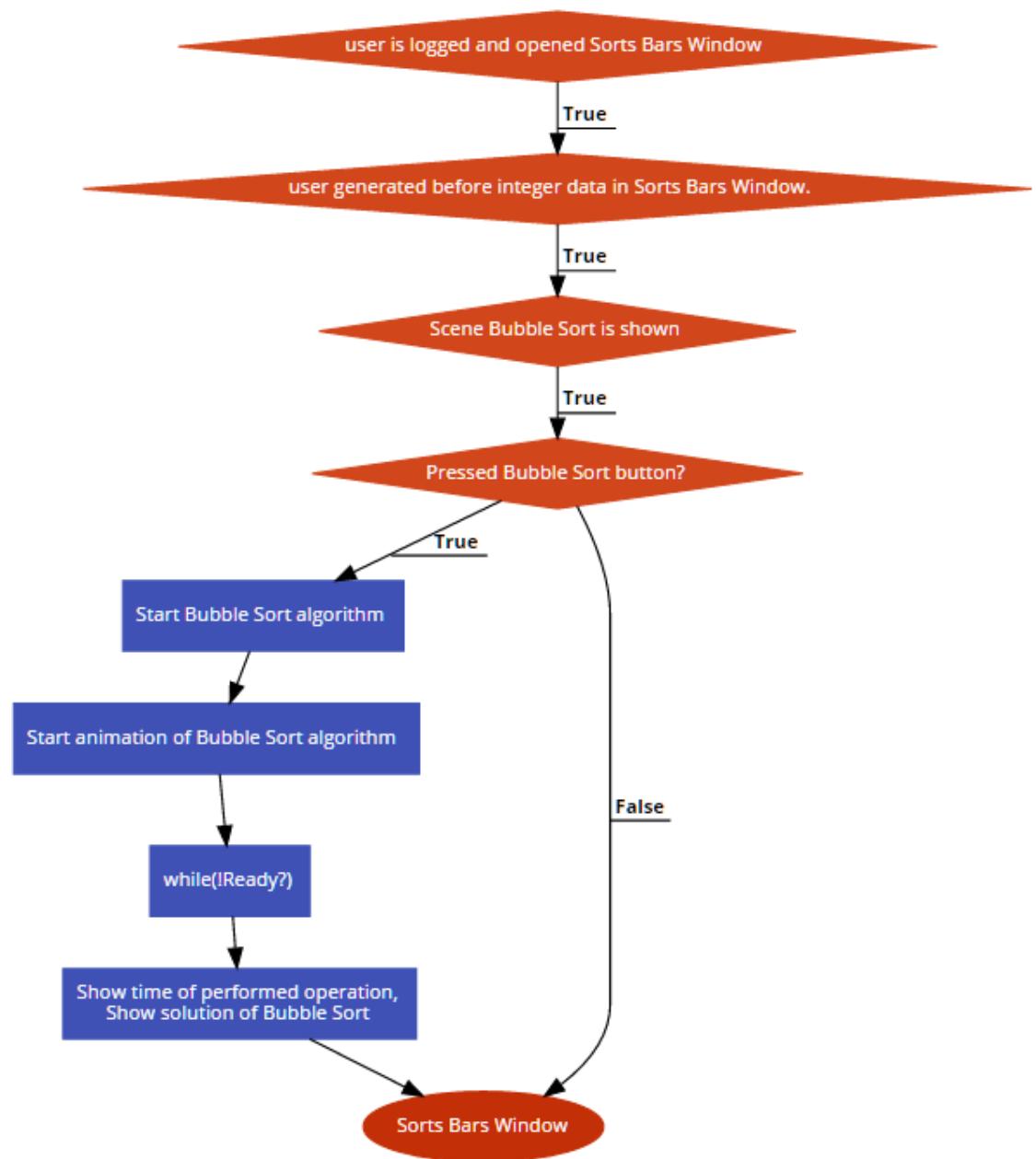


Figure 2 Flow Diagram FR1

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Bubble Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Bubble Sort in Sorts Bars window.
- A2. The system starts Bubble Sort algorithm.
- A3. The system shows animation of Bubble Sort algorithm.
- A4. When the Bubble Sort algorithm ends system shows time of performed Bubble Sort operation.
- A5. When animation is finished system shows final Bubble Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

Termination

The user press Enter Bubble Sort button

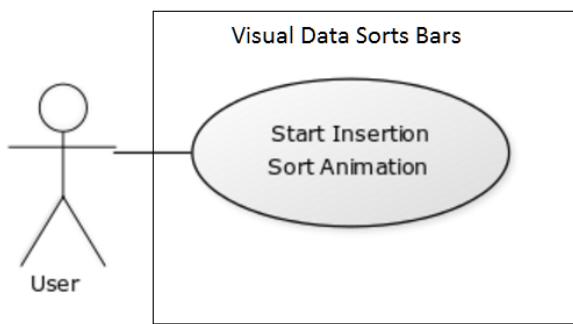
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



4.1.4 Requirement 2: "FR2"



4.1.4.1 Description & Priority

Visual Animation of Insertion Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press

Insertion Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

4.1.4.2 Use Case

Scope

The scope of this use case is to start sort Insertion Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Insertion Sort in Sorts Bars window

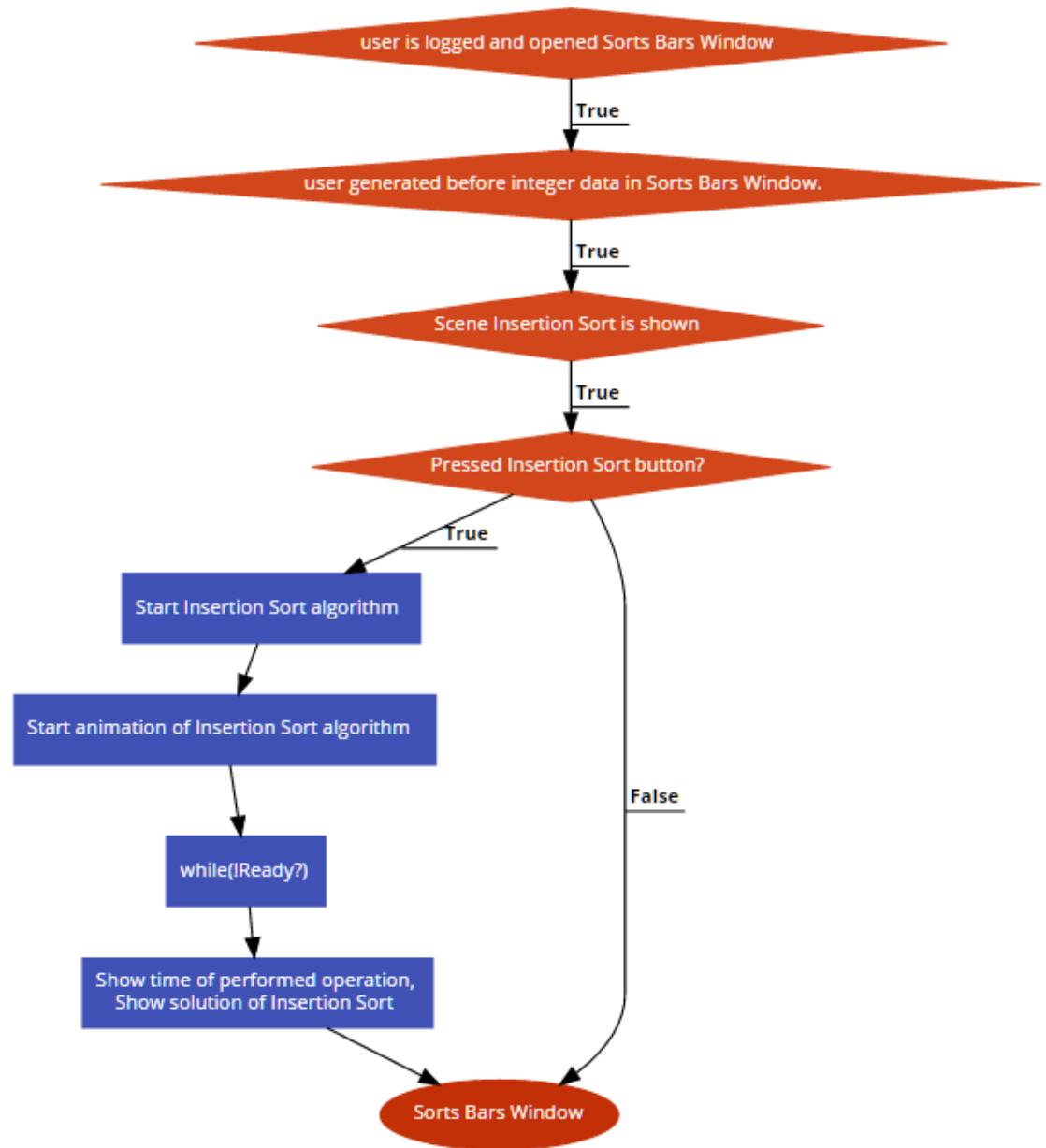


Figure 2 Flow Diagram FR2

Termination Flow Description

Precondition

System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Insertion Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Insertion Sort in Sorts Bars window.
- A2. The system starts Insertion Sort algorithm.
- A3. The system shows animation of Insertion Sort algorithm.
- A4. When the Insertion Sort algorithm ends system shows time of performed Insertion Sort operation.
- A5. When animation is finished system shows final Insertion Sort scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

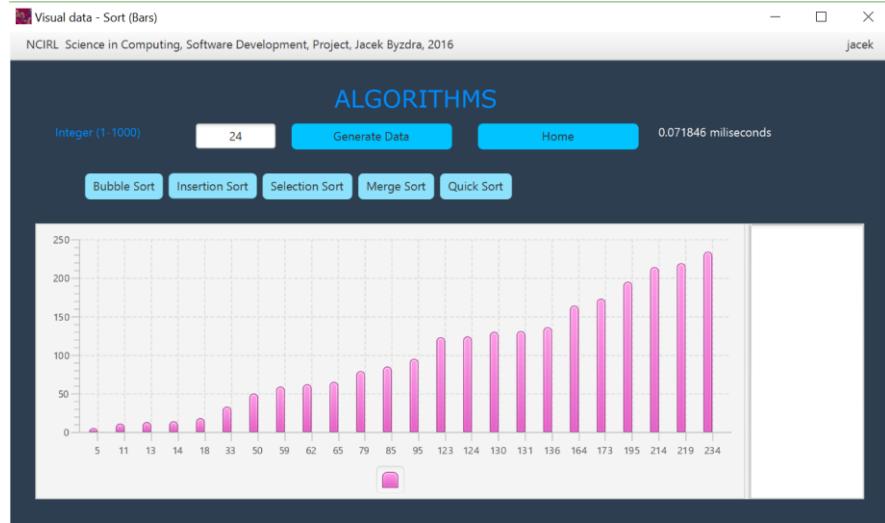
Termination

The user press Enter Insertion Sort button

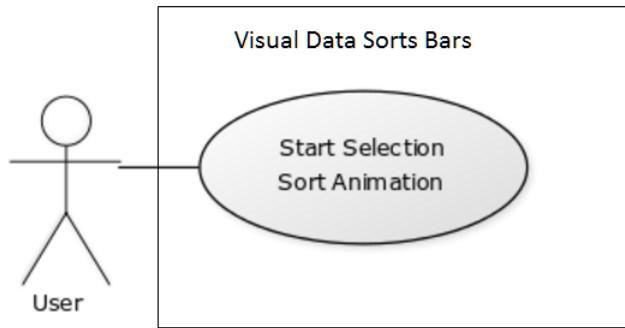
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



4.1.5 Requirement 3:"FR3"



4.1.5.1 Description & Priority

Visual Animation of Selection Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Selection Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

4.1.5.2 Use Case

Scope

The scope of this use case is to start sort Selection Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Selection Sort in Sorts Bars window

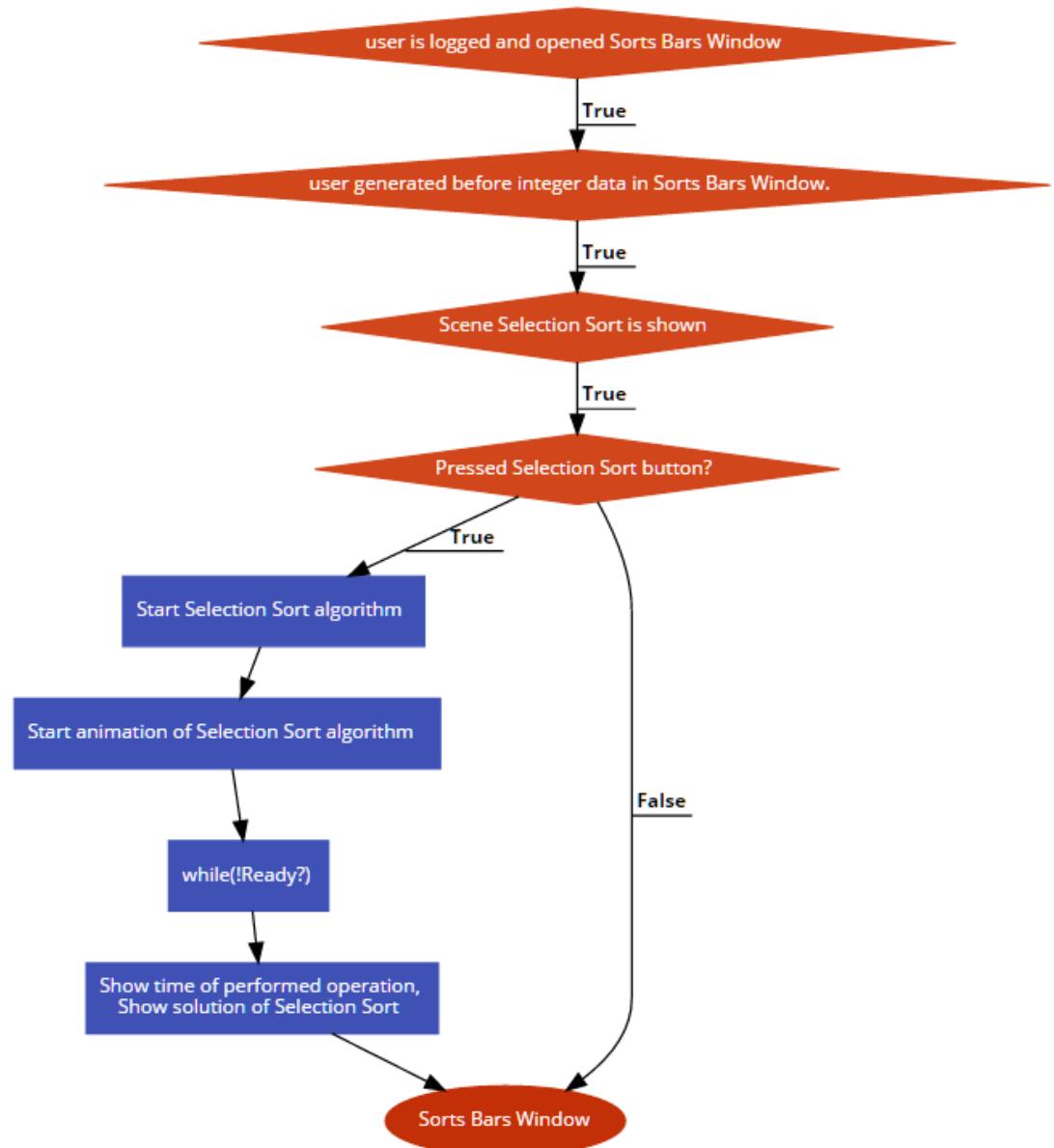


Figure 3 Flow Diagram FR3

Termination Flow Description

Precondition

System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Selection Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Selection Sort in Sorts Bars window.
- A2. The system starts Selection Sort algorithm.
- A3. The system shows animation of Selection Sort algorithm.
- A4. When the Selection Sort algorithm ends system shows time of performed Selection Sort operation.
- A5. When animation is finished system shows final Selection Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

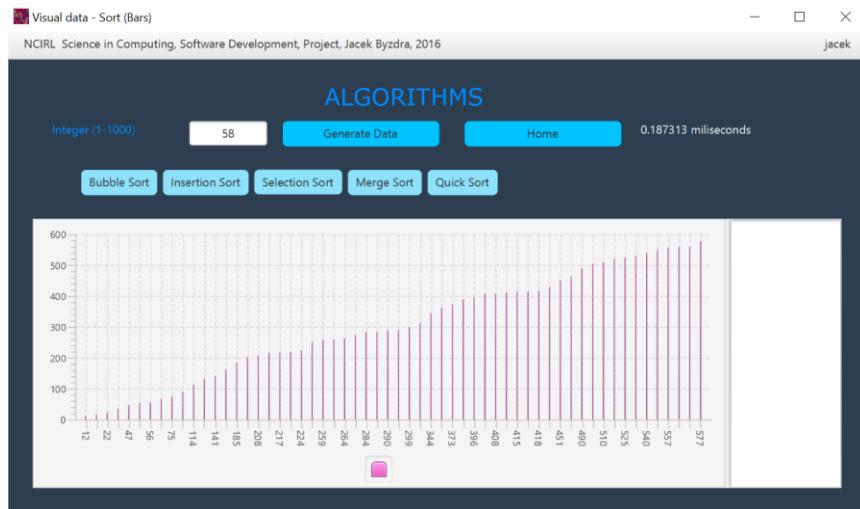
Termination

The user press Enter Selection Sort button

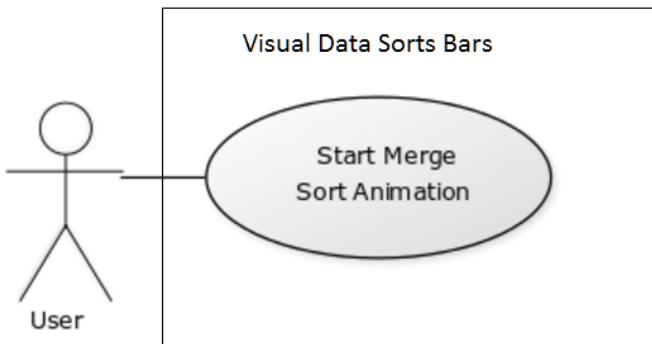
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



4.1.6 Requirement 4: "FR4"



4.1.6.1 Description & Priority

Visual Animation of Merge Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Merge Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

4.1.6.2 Use Case

Scope

The scope of this use case is to start sort Merge Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Merge Sort in Sorts Bars window

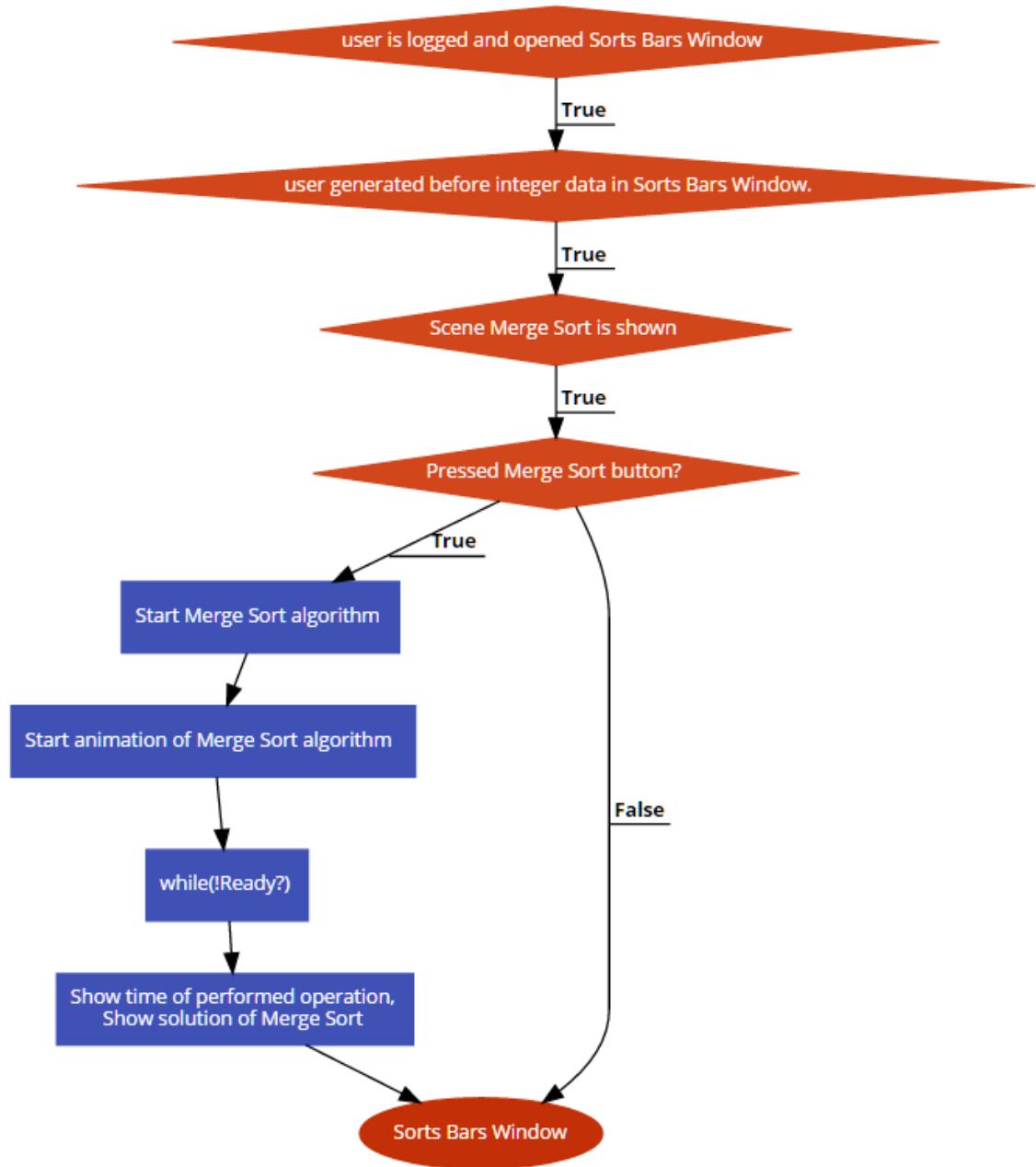


Figure 4 Flow Diagram FR4

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Merge Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Merge Sort in Sorts Bars window.
- A2. The system starts Merge Sort algorithm.
- A3. The system shows animation of Merge Sort algorithm.
- A4. When the Merge Sort algorithm ends system shows time of performed Merge Sort operation.
- A5. When animation is finished system shows final Merge Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

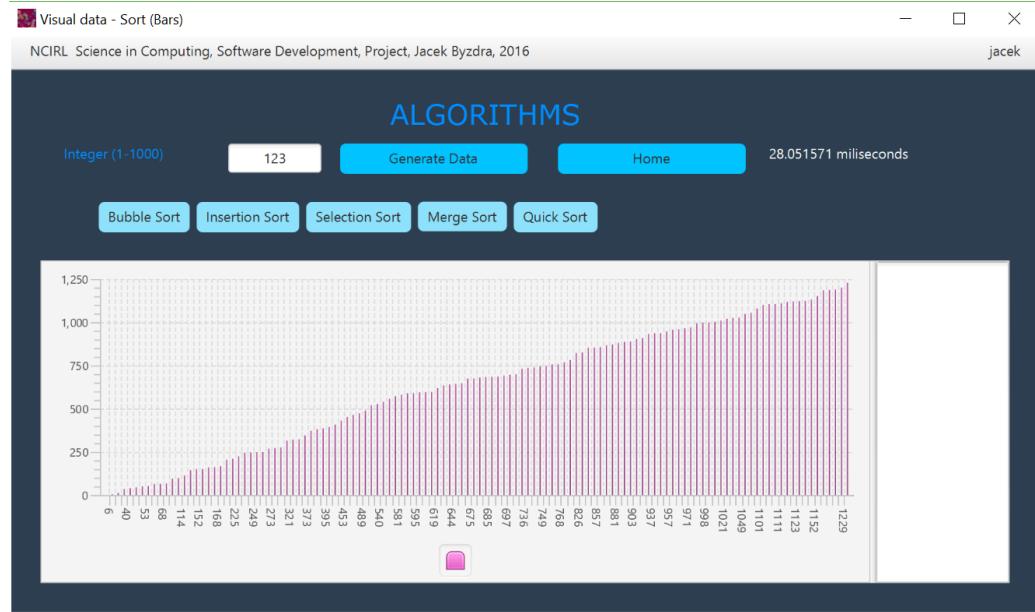
Termination

The user press Enter Merge Sort button

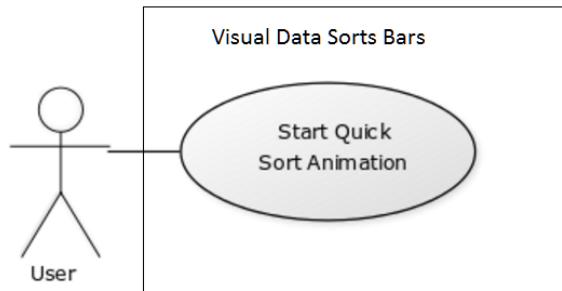
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



4.1.7 Requirement 5:"FR5"



4.1.7.1 Description & Priority

Visual Animation of Quick Sort Algorithm (Sorts Bars View). Show sorting in ascending order. When the program runs in Sorts Bars Window, and user press Quick Sort button. System is running, User is logged in, Sort Bars window is opened, User generated before integer data in Sorts Bars Window.

4.1.7.2 Use Case

Scope

The scope of this use case is to start sort Quick Sort in Sorts Bars window.

Description

This use case describes the process of sorting algorithm Quick Sort in Sorts Bars window

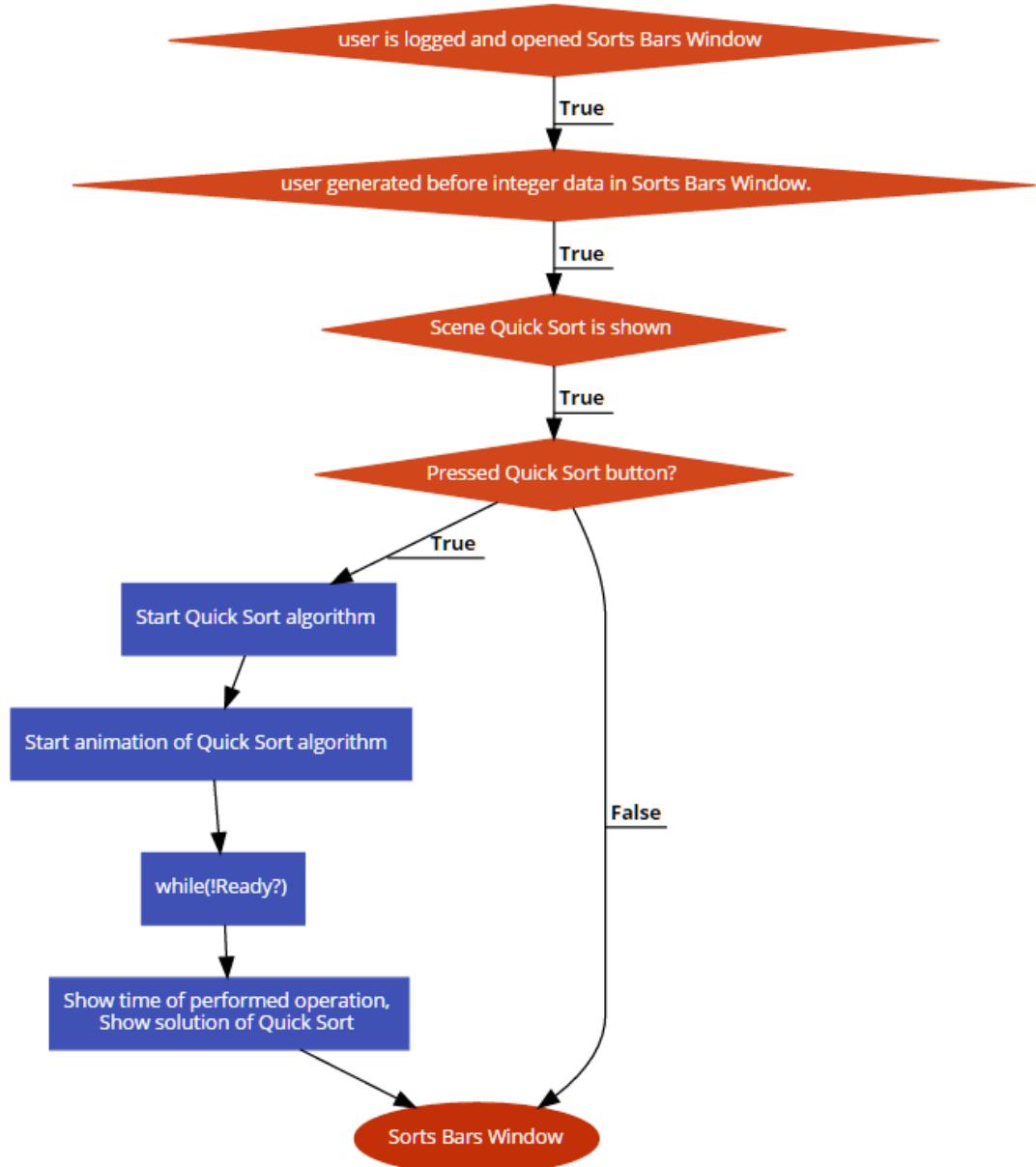


Figure 4 Flow Diagram FR4

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Bars window is opened, User generated before integer data in Sorts Bars Window.

Activation

This use case starts when a user press Quick Sort button.

Main flow

- A1. User is in Sorts Bars Window and generated before integer data in Sorts Bars Window. System shows the scene Quick Sort in Sorts Bars window.
- A2. The system starts Quick Sort algorithm.
- A3. The system shows animation of Quick Sort algorithm.
- A4. When the Quick Sort algorithm ends system shows time of performed Quick Sort operation.
- A5. When animation is finished system shows final Quick Sorts scene in Sorts Bars window with solution.

Alternate flow

- B1: The system shows Sorts Bars window.

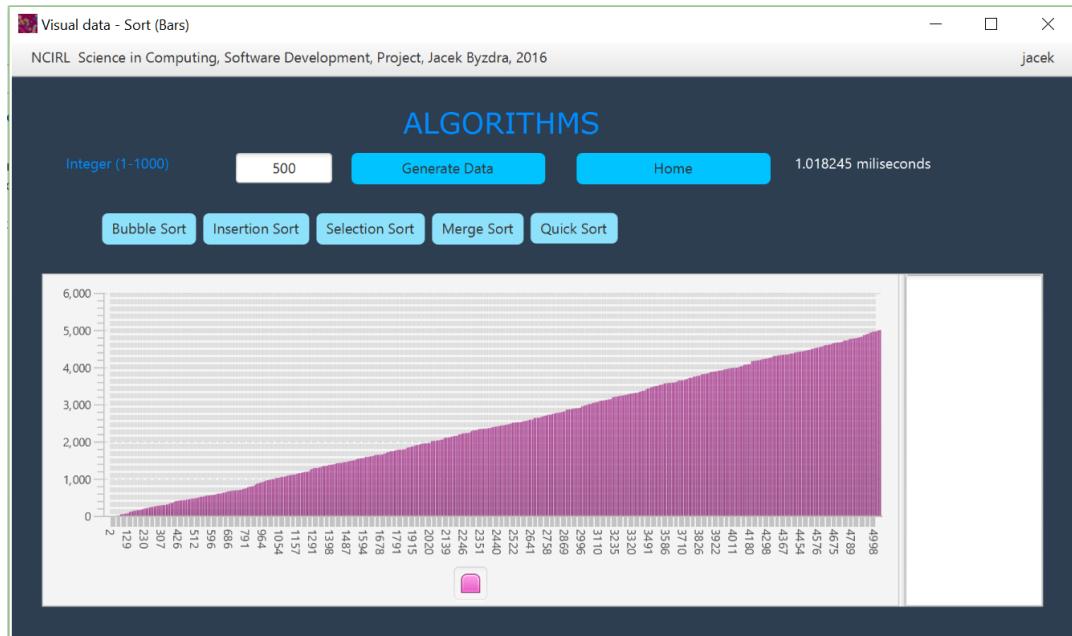
Termination

The user press Enter Quick Sort button

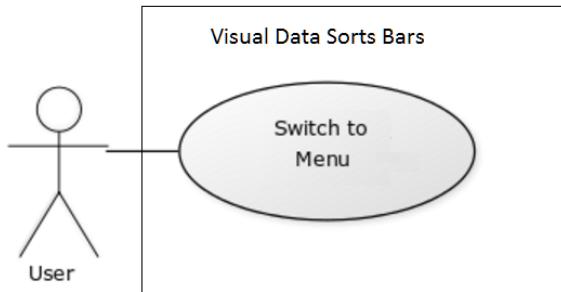
Post condition

The system displays sorted bars in ascending order in Sorts Bars window.

Post condition Mock



4.1.8 Requirement 7:"FR7"



4.1.8.1 Description & Priority

Switch to Menu window from Sort Bars window after home button is pressed. When the program runs and Sorts Bars window is opened. System should switch to main window independently of running sorting algorithms. User is logged in, System is running and Sorts Bars Window is open.

4.1.8.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Sorts Bars window.

Description

This use case describes the process of switching to Menu window from Sorts Bars window.

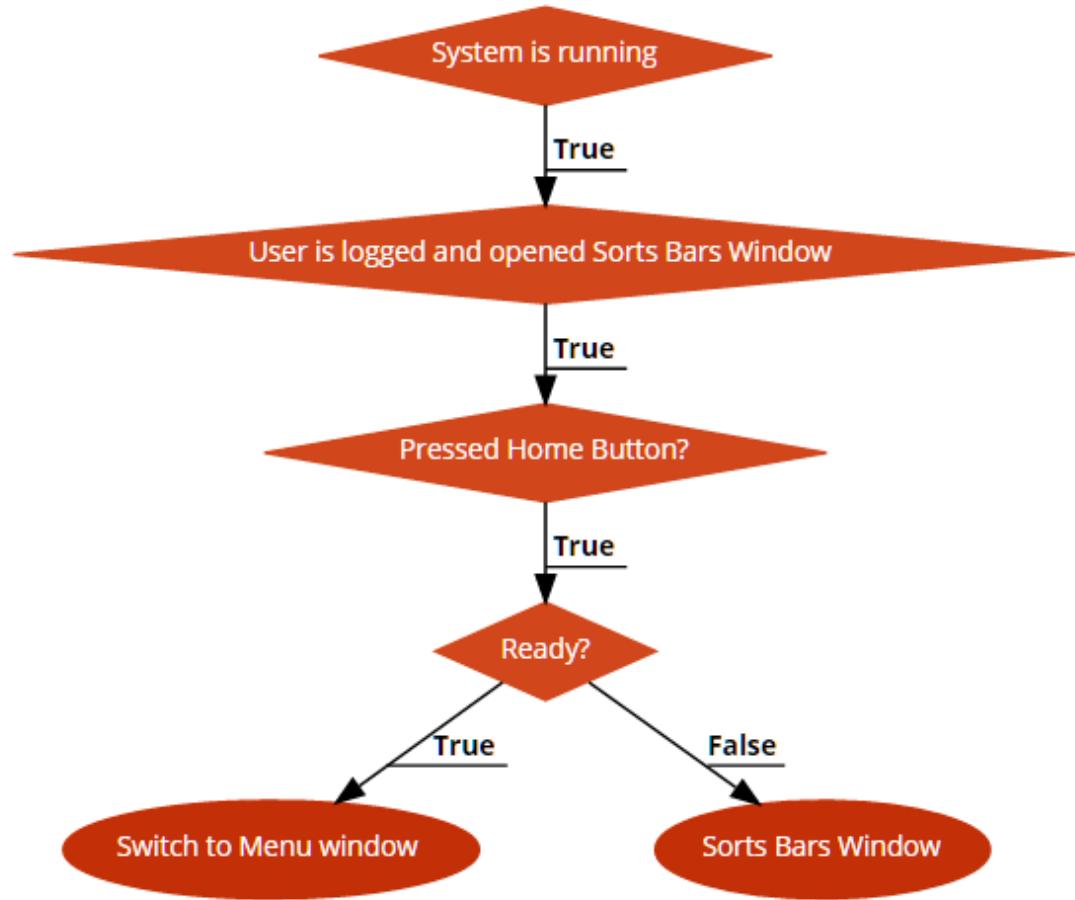


Figure 7 Flow Diagram FR7

Termination Flow Description

Precondition

User is logged in, System is running and Sorts Bars Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1.User is in Sorts Bars Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Sorts Bars window.

Termination

The user press Home button

Post condition

The system goes to Menu window.

Post condition Mock

Visual data
NCIRL Science in Computing, Software Development, Project, Jacek Byzdra, 2016 jacek

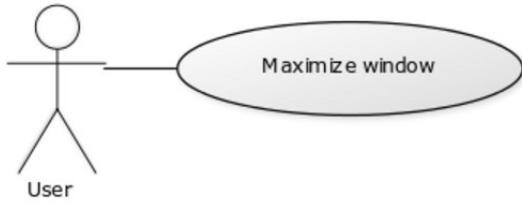
ALGORITHMS

Quiz Description

Hints ▾ Sorts Bars Sorts Panels Trees Graphs

BUBBLE SORT ALGORITHM (BSA) IS BASED ON COMPARISON OF TWO SUBSEQUENT ELEMENTS AND SWAP OF THEM EACH OTHER IF THEY ARE NOT IN QUEUE ORDER. THE TIME COMPLEXITY OF THE ALGORITHM IS AVERAGE $O(N^2)$. THE MEMORY COMPLEXITY OF BSA IS $O(1)$. THE ALGORITHM PERFORMS $N-1$ STEPS, AND IN EACH STEP IT DOES $N-K$ COMPARISON, WHERE K IS EQUAL NUMBER OF THE STEP.	4 2 3 1 0 2 4 3 1 0 2 3 4 1 0 2 3 1 4 0 2 3 1 0 4 2 1 3 0 4 2 1 0 3 4 1 2 0 3 4 1 0 2 3 4 0 1 2 3 4	procedure bubbleSort(A : the number of elements to sort) n = the number of elements(A) do for (i = 0; i < n-1; i++) do: if A[i] > A[i+1] then swap(A[i], A[i+1]) end if end for n = n-1 while n > 1 end procedure
---	--	---

4.1.9 Requirement 8:"FR8"



4.1.9.1 Description & Priority

Maximize Window when maximize button is pressed.

4.1.9.2 Use Case

Scope

The scope of this use case is Maximize Window when user press maximize button.

Description

This use case describes the process of Maximizing window.

Termination Flow Description

Precondition

System is running. User is logged in, Register window is not opened.

Activation

This use case starts when a user press Maximize button.

Main flow

- A1.User press maximize button
- A2. Window is maximize

Alternate flow

- B1: Window size is not changed.

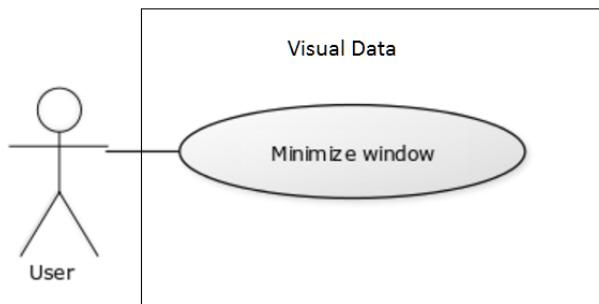
Termination

The user press Maximize button

Post condition

The system Maximize window.

4.1.10 Requirement 9:"FR9"



4.1.10.1 Description & Priority

Minimize Window when minimize button is pressed.

4.1.10.2 Use Case

Scope

The scope of this use case is Minimize Window when user press minimize button.

Description

This use case describes the process of Minimizing window.

Termination Flow Description

Precondition

System is running. User is logged in, Register window is not opened.
Window is in maximum size.

Activation

This use case starts when a user press Minimize button.

Main flow

- A1. User press minimize button
- A2. Window is minimized

Alternate flow

- B1: Window size is not changed.

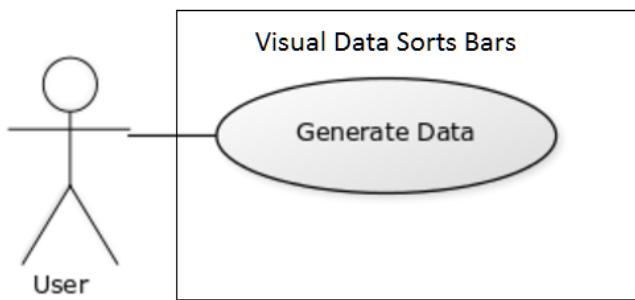
Termination

The user press Minimize button

Post condition

The system Minimize window.

4.1.11 Requirement 10:"FR10"



4.1.11.1 Description & Priority

Generate Data for sorting bars when the Generate Data button is pressed. When the program runs and Sorts Bars window is open, user provided correct integer number to sort before, and user pressed generate data button. System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label. After Generate Data button is pressed

System generates the number of bars equal to generated integer number and displays the bars in the scene.

4.1.11.2 Use Case

Scope

The scope of this use case is Generate Data when user press Generate Data Button in Sorts Bars window.

Description

This use case describes the process of Generating Data when user press Generate Data Button in Sorts Bars window.

Termination Flow Description

Precondition

System is running, User is logged in, and Sorts Bars Window is open, user provided correct integer number to sort label.

Activation

This use case starts when a user press Generate Data button.

Main flow

- A1. User press Generate Data button
- A2. System generates the number of bars equal to generated integer number and displays the bars in the scene

Alternate flow

- B1: System keeps the number of bars equal to default number 120.

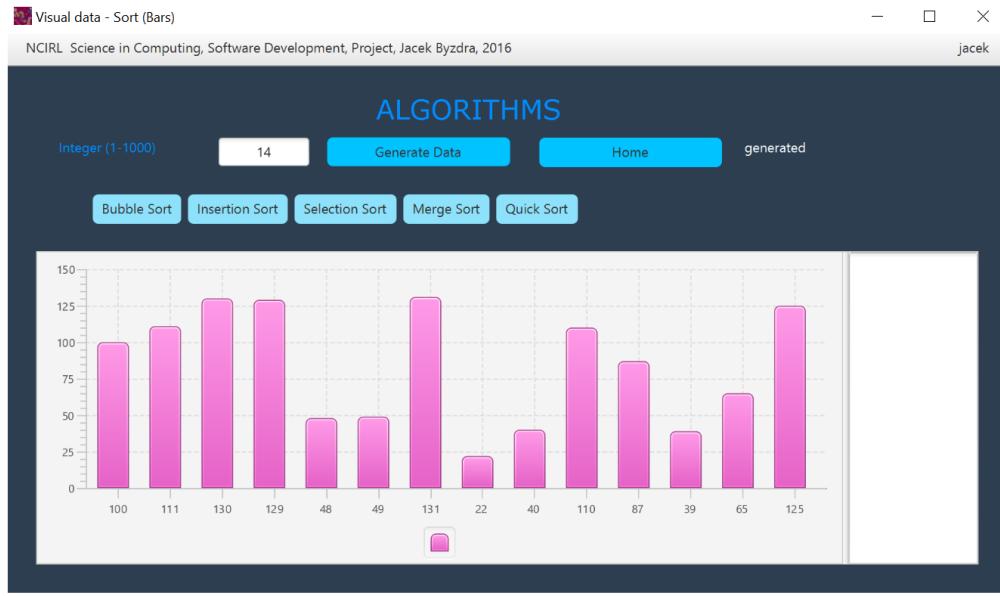
Termination

The user press Generate Data button

Post condition

System generates the number of bars equal to generated integer number and displays the bars in the scene.

Post condition Mock



4.1.12 Requirement 11:"FR11"

4.1.12.1 Description & Priority

Display information "Incorrect sort number". When the program runs and Sorts Bars window is open, user provided incorrect integer number to sort before, and user pressed generate data button. System is running, user is logged in and Sorts Bars Window is open, user provided incorrect integer number to sort label.

4.1.12.2 Use Case

Scope

The scope of this use case is Display information "Incorrect sort number". When the program runs and Sorts Bars window is open, user

provided incorrect integer number to sort before, and user pressed generate data button.

Description

This use case describes the process of displaying information "Incorrect sort number".

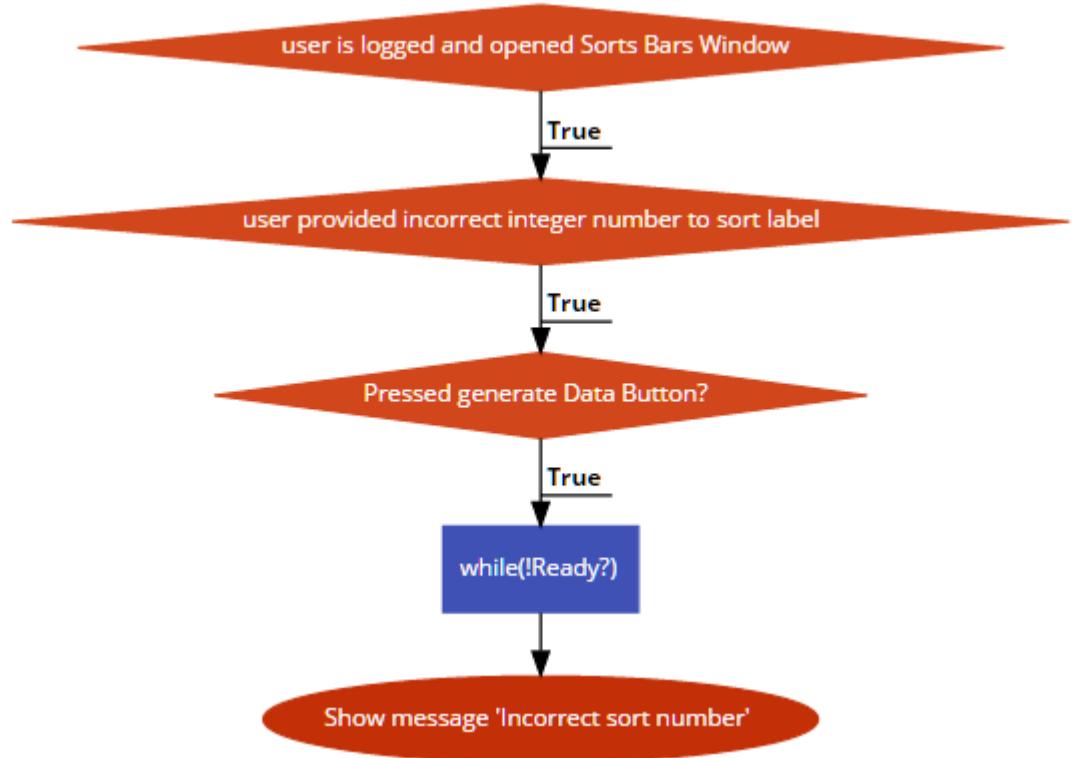


Figure 11: Flow Diagram FR 11

Termination Flow Description

Precondition

System is running, user is logged in and Sorts Bars Window is open, user provided incorrect integer number to sort label.

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

A1. User press Generate Data button

A2. System displays information "Incorrect sort number".

Alternate flow

B1: Button Generate Data is not pressed.

Termination

The user press Generate Data button after incorrect number was provided to sort label.

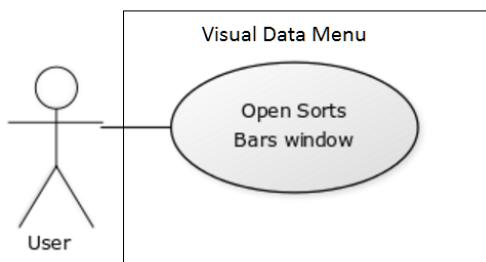
Post condition

The system displays information "Incorrect sort number".

Post condition Mock



4.1.13 Requirement 12:"FR12"



4.1.13.1 Description & Priority

Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.
When the program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

4.1.13.2 Use Case

Scope

The scope of this use case is Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window. When the program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

Description

This use case describes the process of displaying Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Termination Flow Description

Precondition

Program runs in Sorts Bars Window, System is running , user is logged in and the Sorts Bars window is opened.

Activation

This use case starts when Sorts Bars window is opened.

Main flow

- A1. User opens Sorts Bars window
- A2. System Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Alternate flow

- B1: Sort Bars window is not opened.

Termination

The user opens Sorts Bars window.

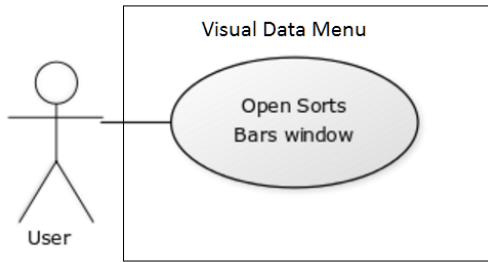
Post condition

The system Display Buttons Generate Data, Home, Integer Sort Label in Sorts Bars window.

Post condition Mock



4.1.14 Requirement 13:"FR13"



4.1.14.1 Description & Priority

Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window. When the program runs in Sorts Bars window, and user entered and generated correct integer number to sort. When the program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort.

4.1.14.2 Use Case

Scope

The scope of this use case is Display Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Description

This use case describes the process of displaying Button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Termination Flow Description

Precondition

Program runs, user is logged in in Sorts Bars window, and user entered and generated correct integer number to sort.

Activation

This use case starts when user generated correct integer number to sort.

Main flow

- A1. User generates correct integer number to sort
- A2. System displays button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Alternate flow

- B1: user not generates number to sort.

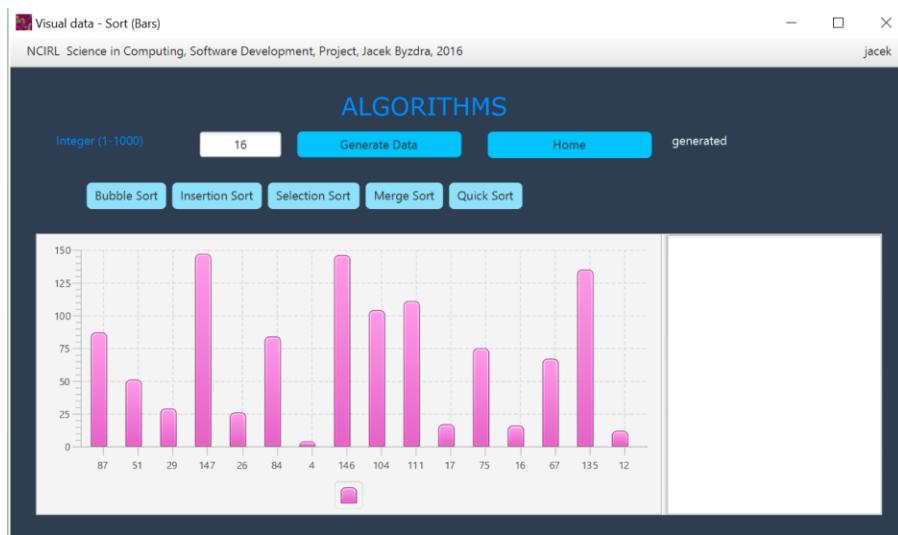
Termination

User generates correct integer number to sort.

Post condition

System displays button: Bubble Sort, Insertion Sort, Selection Sort, merge Sort, Quick Sort in Sorts Bars window.

Post condition Mock



4.1.15 Requirement 14:"FR14"

4.1.15.1Description & Priority

Display information about logged user in Sorts Bars window. When the program runs , user is logged in, and Sorts Bars window is opend.

4.1.15.2Use Case

Scope

The scope of this use case is Display information about logged user in Sorts Bars window. When the program runs , user is logged in, and Sorts Bars window is opend.

Description

This use case describes the process of displaying information about logged user in Sorts Bars window.

Termination Flow Description

Precondition

Program runs , user is logged in, and Sorts Bars window is opend.

Activation

This use case starts when user user is logged in, and Sorts Bars window is opend.

Main flow

- A1. After login user is directed to Sorts Bars window
- A2. System displays information about logged user in right top corner.

Alternate flow

- B1: user is not logged to system.

Termination

User successfully is logged to system.

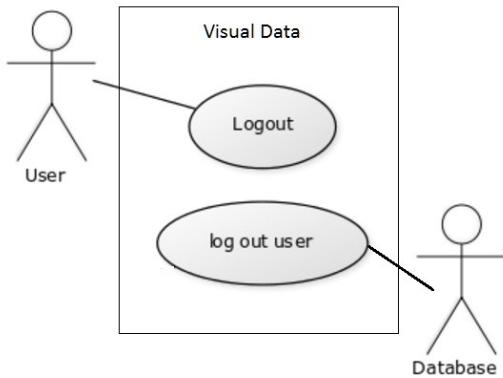
Post condition

System displays information about logged user.

Post condition Mock



4.1.16 Requirement 15:"FR15"



4.1.16.1 Description & Priority

Log user out after Logout button is pressed. When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button. When the program runs , user is logged in, and Sorts Bars window is opend, and user pressed Logout button.

4.1.16.2 Use Case

Scope

The scope of this use case is Log user out after Logout button is pressed.

Description

This use case describes the process of logging user out after Logout button is pressed.

Termination Flow Description

Precondition

Program runs , user is logged in, and Sorts Bars window is opend.

Activation

User press Logout button.

Main flow

A1. User press logout button

A2. System logout user.

Alternate flow

B1: user is logged in.

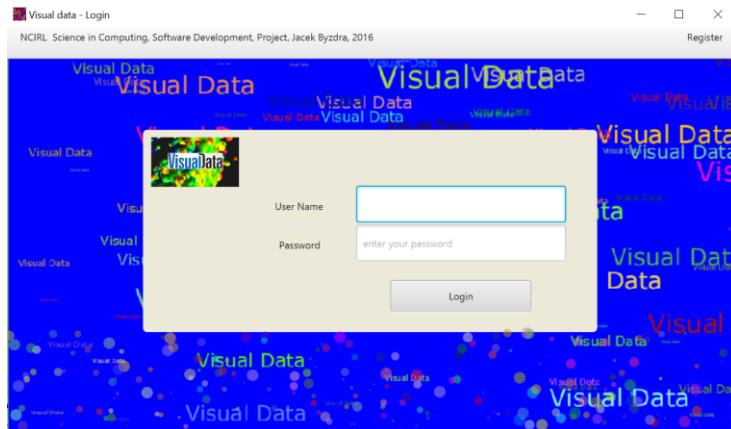
Termination

User successfully is logged out from system.

Post condition

System switch user to Login window.

Post condition Mock



4.1.17 Requirement 16:"FR16"

4.1.17.1 Description & Priority

Visual Animation of Bubble Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user

press Bubble Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

4.1.17.2 Use Case

Scope

The scope of this use case is Visual Animation of Bubble Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Bubble Sort Algorithm (Sorts Panels View).

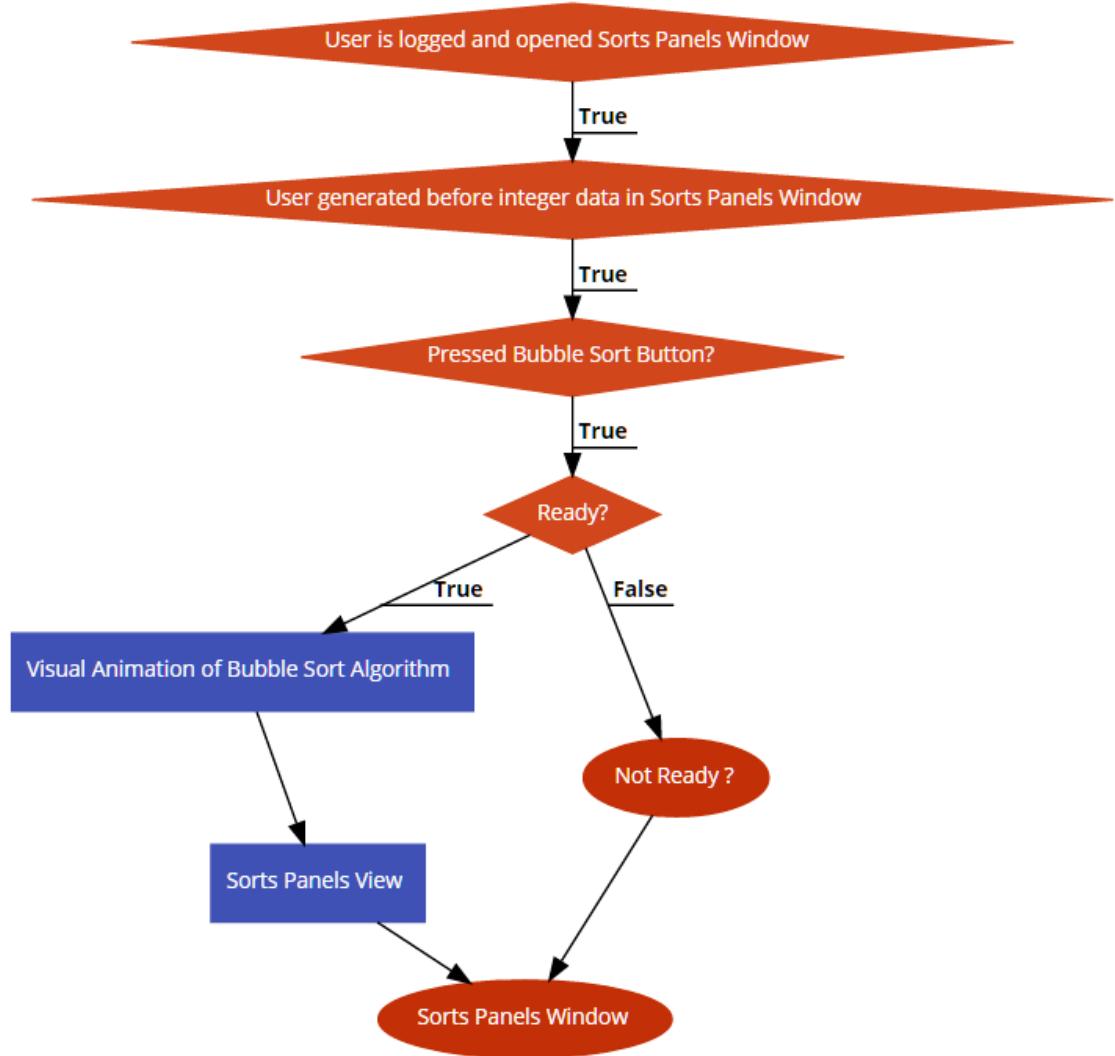


Figure 16: Flow Diagram FR16

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Bubble Sort button.

Main flow

- A1. User press Bubble sort button
- A2. System animates Bubble Sort algorithm.

Alternate flow

- B1: Bubble Sort button is not pressed.

Termination

User press Bubble sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

4.1.18 Requirement 17:"FR17"

4.1.18.1 Description & Priority

Visual Animation of Insertion Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Insertion Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

4.1.18.2 Use Case

Scope

The scope of this use case is Visual Animation of Insertion Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Insertion Sort Algorithm (Sorts Panels View).

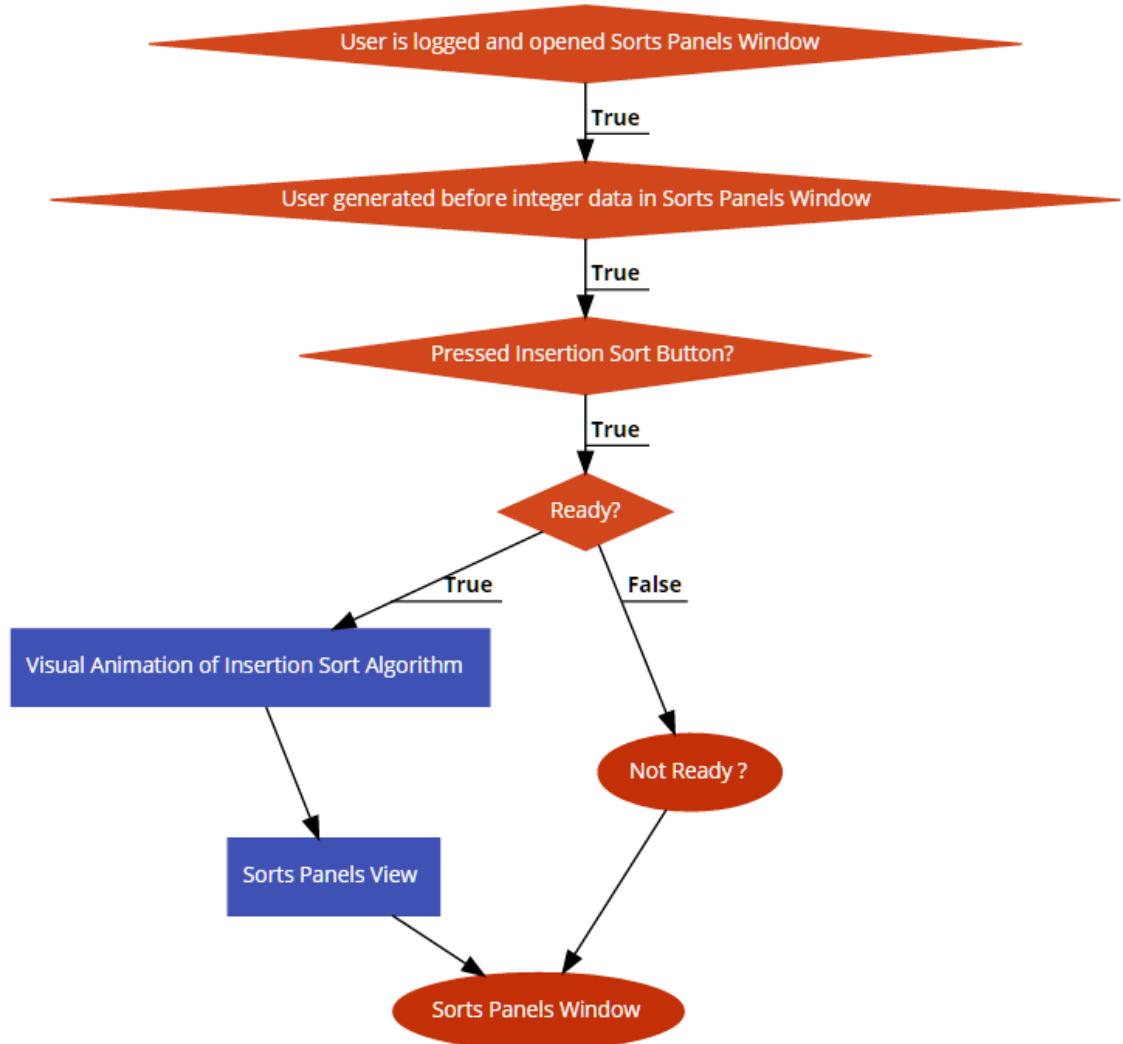


Figure 17: Flow Diagram FR17

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Insertion Sort button.

Main flow

- A1. User press Insertion sort button
- A2. System animates Insertion Sort algorithm.

Alternate flow

- B1: Insertion Sort button is not pressed.

Termination

User press Insertion sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

4.1.19 Requirement 18:"FR18"

4.1.19.1 Description & Priority

Visual Animation of Selection Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Selection Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

4.1.19.2 Use Case

Scope

The scope of this use case is Visual Animation of Selection Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Selection Sort Algorithm (Sorts Panels View).

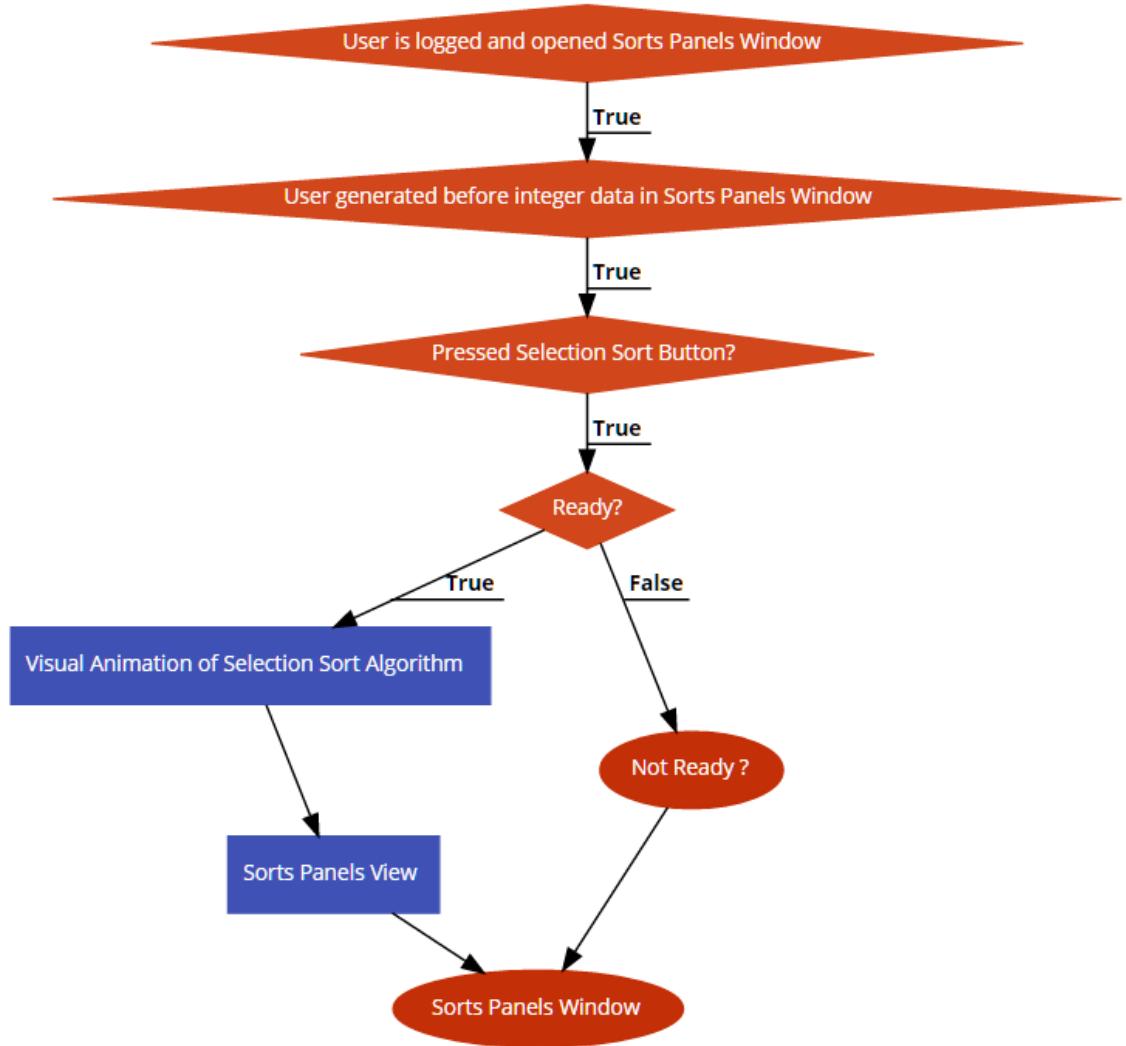


Figure 18: Flow Diagram FR18

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Selection Sort button.

Main flow

- A1. User press Selection sort button
- A2. System animates Selection Sort algorithm.

Alternate flow

- B1: Selection Sort button is not pressed.

Termination

User press Selection sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

4.1.20 Requirement 19:"FR19"

4.1.20.1 Description & Priority

Visual Animation of Heap Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Heap Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

4.1.20.2 Use Case

Scope

The scope of this use case is Visual Animation of Heap Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Heap Sort Algorithm (Sorts Panels View).

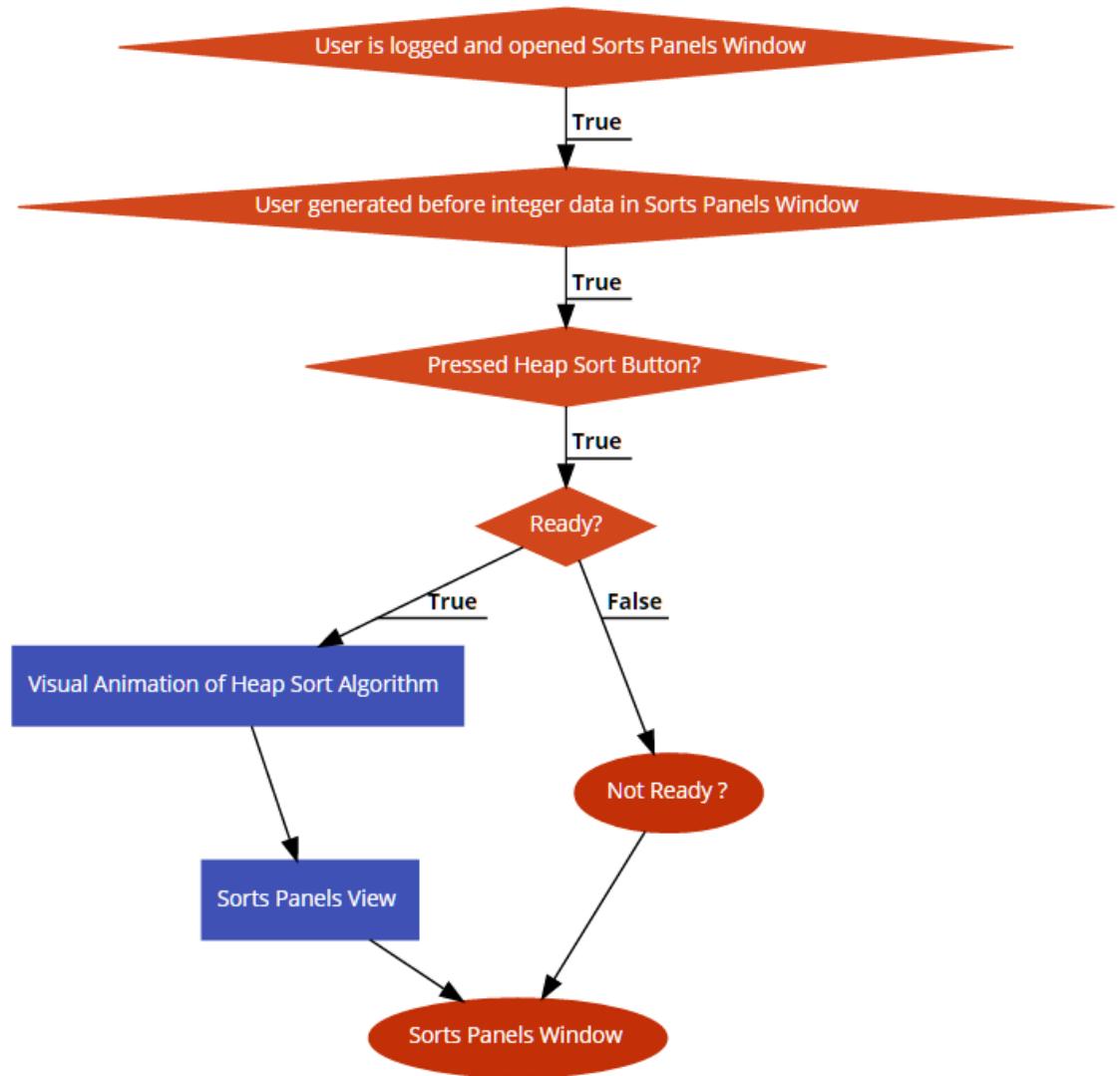


Figure 19: Flow Diagram FR19

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Heap Sort button.

Main flow

- A1. User press Heap sort button
- A2. System animates Heap Sort algorithm.

Alternate flow

B1: Heap Sort button is not pressed.

Termination

User press Heap sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

4.1.21 Requirement 20:"FR20"

4.1.21.1 Description & Priority

Visual Animation of Quick Sort Algorithm (Sorts Panels View). Show sorting in ascending order. When the program runs in Sorts Panels Window, and user press Quick Sort button System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

4.1.21.2 Use Case

Scope

The scope of this use case is Visual Animation of Quick Sort Algorithm (Sorts Panels View).

Description

This use case describes the process of Visual Animation of Quick Sort Algorithm (Sorts Panels View).

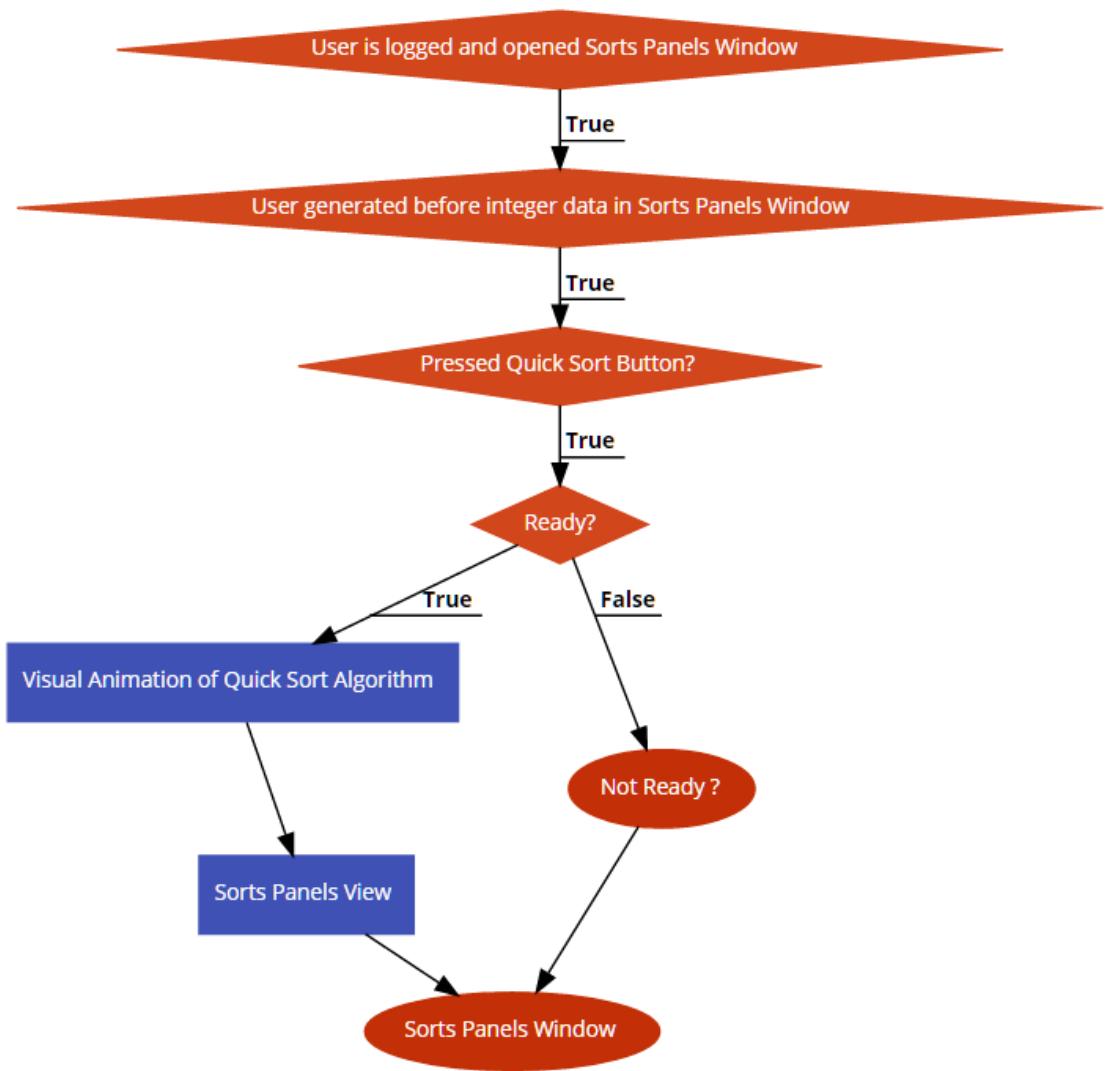


Figure 20: Flow Diagram FR20

Termination Flow Description

Precondition

System is running, User is logged in, Sorts Panels window is opened, User generated before integer data in Sorts Panels Window.

Activation

User press Quick Sort button.

Main flow

- A1. User press Quick sort button
- A2. System animates Quick Sort algorithm.

Alternate flow

B1: Quick Sort button is not pressed.

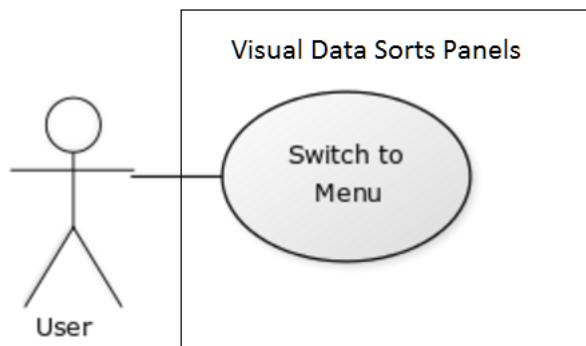
Termination

User press Quick sort button.

Post condition

System displays sorted in ascending order panels in Sorts Panels window.

4.1.22 Requirement 21:"FR21"



4.1.22.1 Description & Priority

Switch to Menu window from Sorts Panels window after home button is pressed.
When the program runs and Sorts Panels window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Sorts Panels Window is open.

4.1.22.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Sorts Panels window.

Description

This use case describes the process of switching to Menu window from Sorts Panels window.

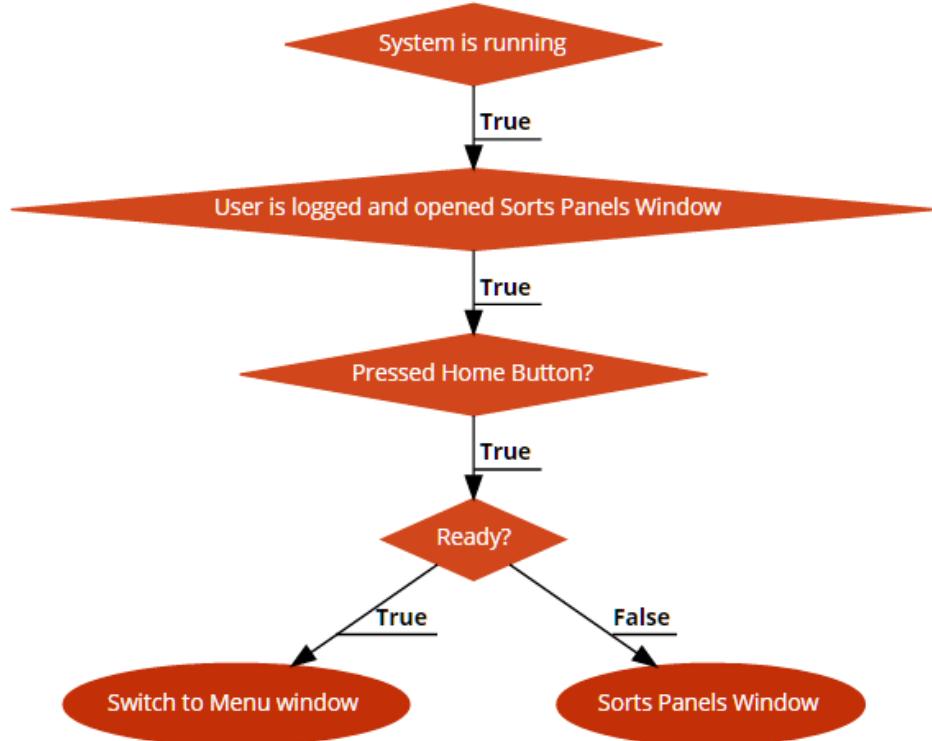


Figure 21 Flow Diagram FR21

Termination Flow Description

Precondition

User is logged in, System is running and Sorts Panels Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1.User is in Sorts Panels Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Sorts Panels window.

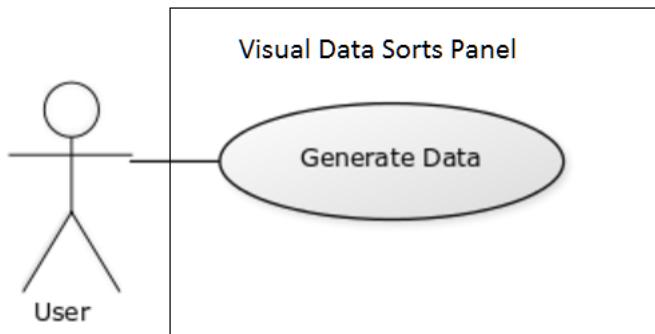
Termination

The user press Home button

Post condition

The system goes to Menu window.

4.1.23 Requirement 22:"FR22"



4.1.23.1 Description & Priority

Generate Integer Data for sorting bars when the Generate Data button is pressed. When the program runs and Sorts Panels window is open, user provided correct integer number to sort before, and user pressed generate data button. System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label.

4.1.23.2 Use Case

Scope

The scope of this use case is to Generate Integer Data for sorting bars when the Generate Data button is pressed.

Description

This use case describes the process of Generating Integer Data for sorting bars when the Generate Data button is pressed.

Figure 21 Flow Diagram FR21

Termination Flow Description

Precondition

System is running, User is logged in, and Sorts Panels Window is open, user provided correct integer number to sort label.

Activation

This use case starts when a user press Generate Data button.

Main flow

A1. User is in Sorts Panels Window and press Generate Data button.

A2. System generates and draws the number of panels equal the number defined by user

Alternate flow

B1: The system Generate Data Button is not pressed.

Termination

The user press Generate Data button

Post condition

The system display in the scene the number of panels equal the number defined by user.

Post Mock-up



4.1.24 Requirement 23:"FR23"

4.1.24.1 Description & Priority

Display information "Incorrect sort number". When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button. System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label.

4.1.24.2 Use Case

Scope

The scope of this use case is Display information "Incorrect sort number". When the program runs and Sorts Panels window is open, user provided incorrect integer number to sort before, and user pressed generate data button.

Description

This use case describes the process of displaying information "Incorrect sort number".

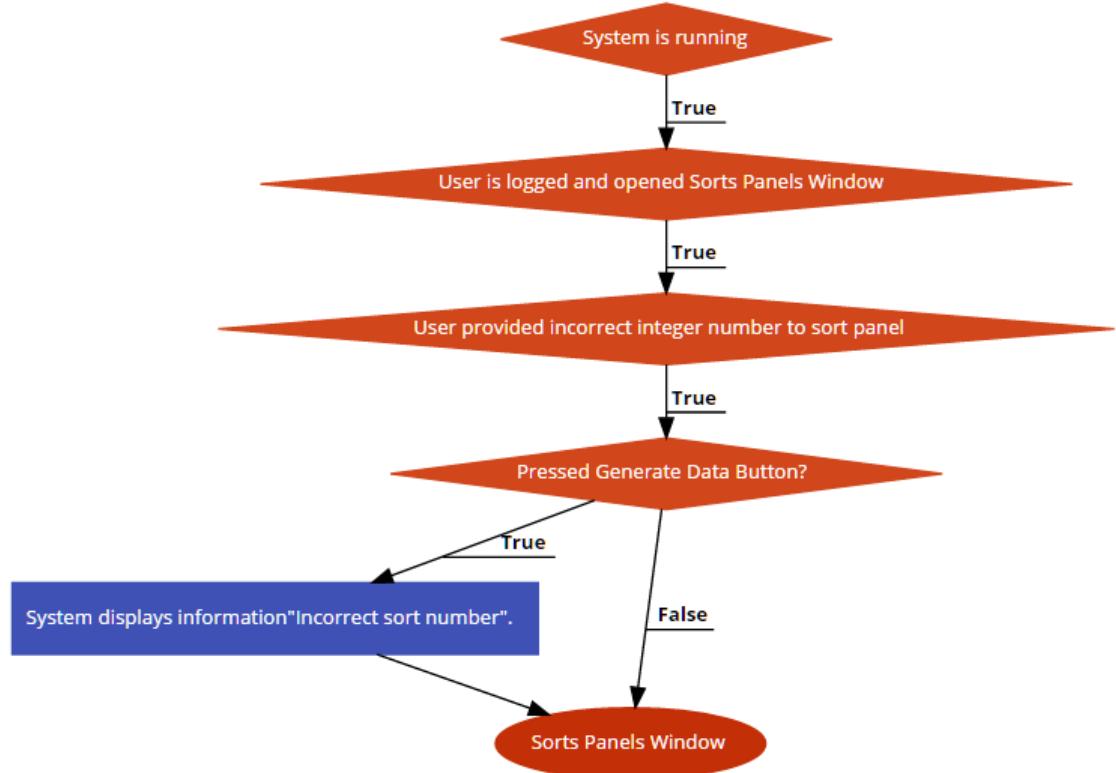


Figure 23: Flow Diagram FR 23

Termination Flow Description

Precondition

System is running, user is logged in and Sorts Panels Window is open, user provided incorrect integer number to sort label.

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

- A1. User press Generate Data button
- A2. System displays information "Incorrect sort number".

Alternate flow

B1: Button Generate Data is not pressed.

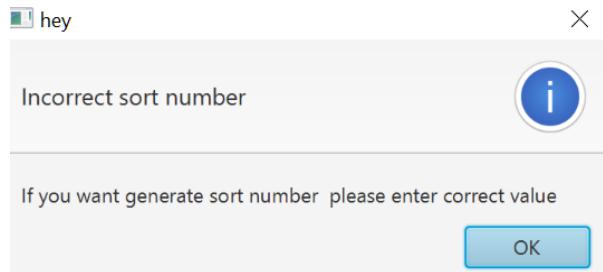
Termination

The user press Generate Data button after incorrect number was provided to sort label.

Post condition

The system displays information "Incorrect sort number".

Post condition Mock



4.1.25 Requirement 40:"FR40"

4.1.25.1 Description & Priority

Display information "Enter correct number of nodes". When the program runs in Binary Search Tree window, user provided before incorrected number of nodes to generate, and pressed Generate Data button. System is running, user is logged in, Binary Search Tree window is opened , user provided incorrect number to BST search label.

4.1.25.2 Use Case

Scope

The scope of this use case is Display information "Enter correct number of nodes". When the program runs in Binary Search Tree window, user

provided before incorrected number of nodes to generate, and pressed Generate Data button.

Description

This use case describes the process of displaying information "Enter correct number of nodes".

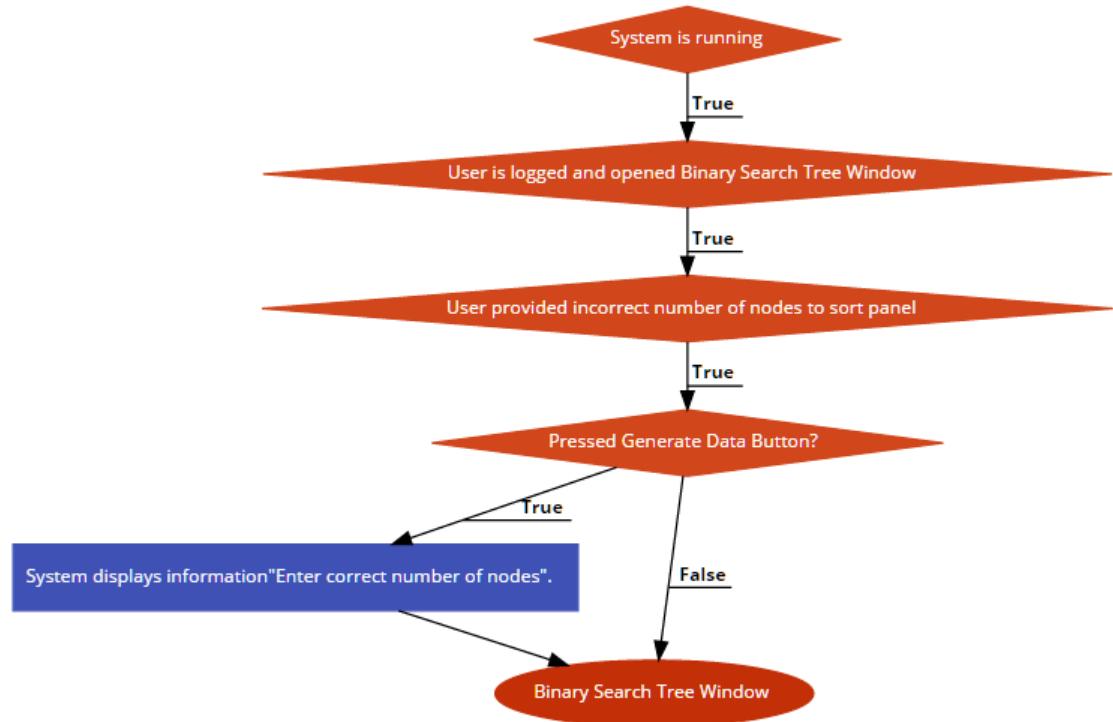


Figure 40: Flow Diagram FR 40

Termination Flow Description

Precondition

System is running, user is logged in, Binary Search Tree window is opened, user provided incorrect number to BST search label .

Activation

This use case starts when a user press Generate Data after incorrect number was provided to sort label.

Main flow

A1. User press Generate Data button

A2. System displays information "Enter correct number of nodes".

Alternate flow

B1: Button Generate Data is not pressed.

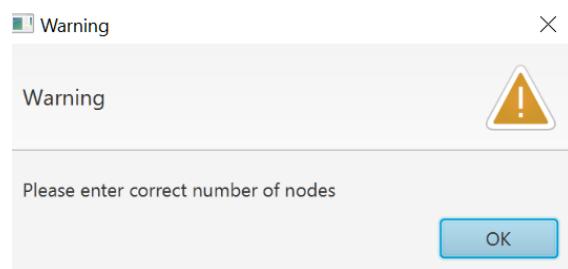
Termination

The user press Generate Data button after provided incorrected number of nodes to sort label.

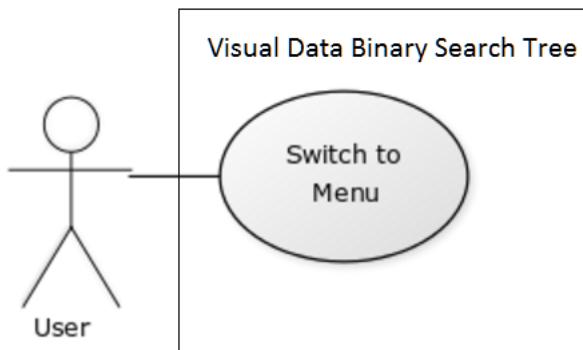
Post condition

The system displays information " Enter correct number of nodes ".

Post condition Mock



4.1.26 Requirement 44:"FR44"



4.1.26.1 Description & Priority

Switch to Menu window from Binary Search Tree window after home button is pressed. When the program runs and Binary Search Tree window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Binary Search Tree Window is open.

4.1.26.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Binary Search Tree window.

Description

This use case describes the process of switching to Menu window from Binary Search Tree window.

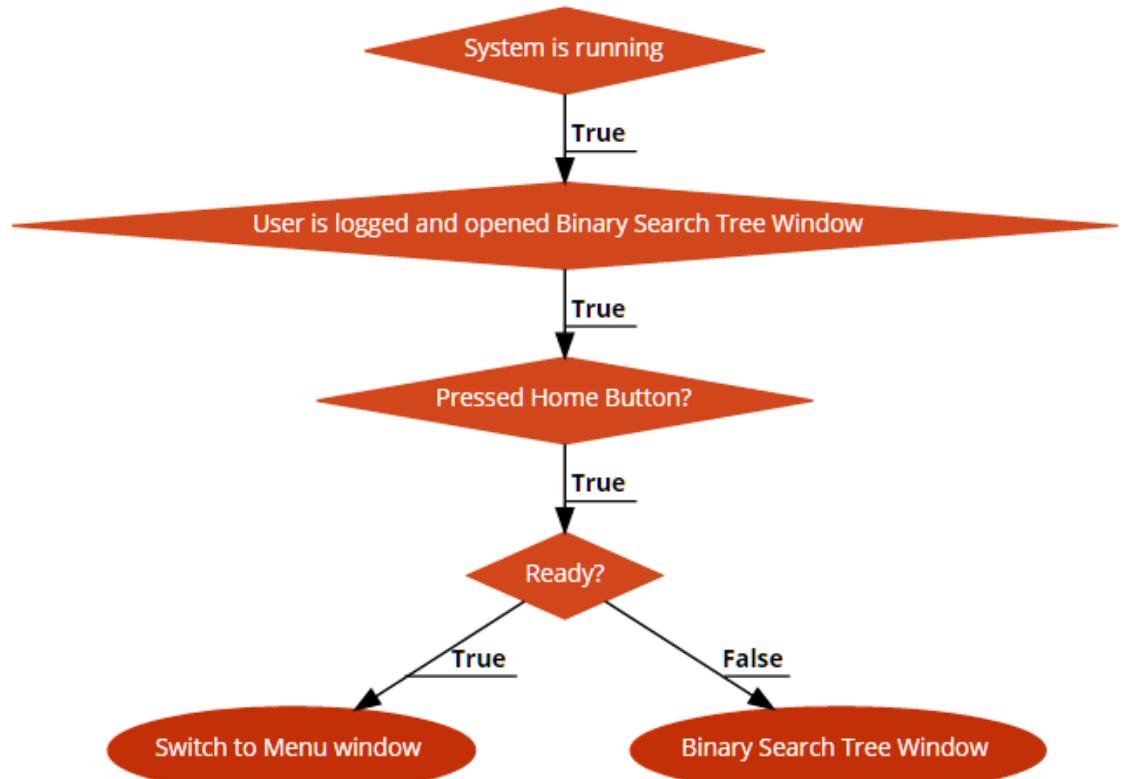


Figure 44 Flow Diagram FR44

Termination Flow Description

Precondition

User is logged in, System is running and Binary Search Tree Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1. User is in Binary Search Tree Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Binary Search Tree window.

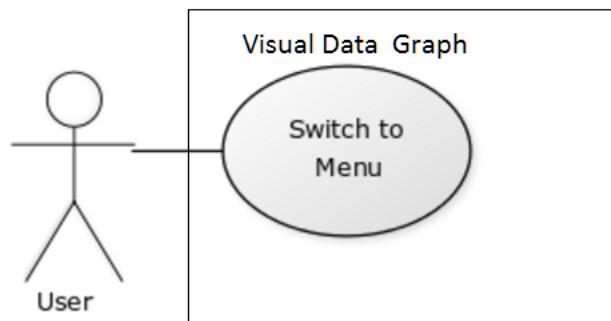
Termination

The user press Home button

Post condition

The system goes to Menu window.

4.1.27 Requirement 46:"FR46"



4.1.27.1 Description & Priority

Switch to Menu window from Graphs window after home button is pressed. When the program runs and Graphs window is opened. System should switch to main window independently of running sorting algorithms User is logged in, System is running and Graphs Window is open.

4.1.27.2 Use Case

Scope

The scope of this use case is to Switch to Menu window from Graphs window.

Description

This use case describes the process of switching to Menu window from Graphs window.

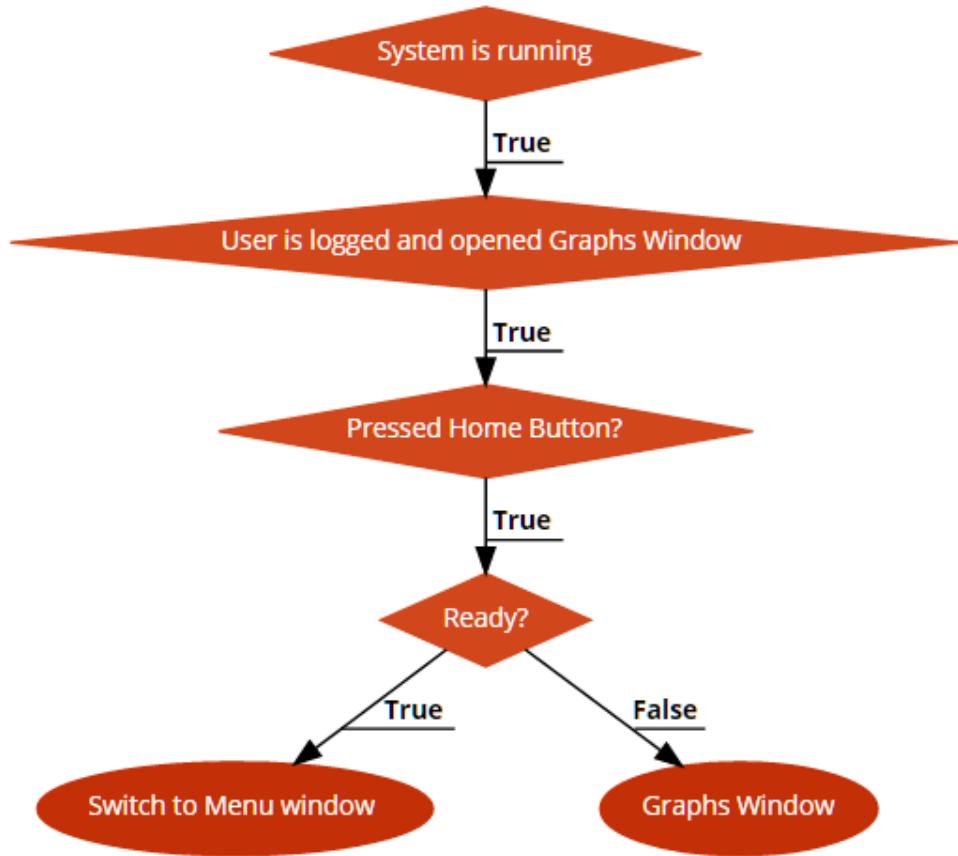


Figure 46 Flow Diagram FR46

Termination Flow Description

Precondition

User is logged in, System is running and Graphs Window is open..

Activation

This use case starts when a user press home button.

Main flow

A1. User is in Graphs Window and press Home button.

A2. System switch to Menu window

Alternate flow

B1: The system shows Graphs window.

Termination

The user press Home button

Post condition

The system goes to Menu window.

4.1.28 Requirement 48:"FR48"

4.1.28.1 Description & Priority

Display information "Incorrect graph size If you want generate graph please enter correct graph size" When the program runs and Graph window is open, user provided incorrect integer number to create graph before, and user pressed Generate Graph button. System is running, user is logged, Graph Window is open, user provided incorrect integer number to graph label.

4.1.28.2 Use Case

Scope

The scope of this use case is Display information information "Incorrect graph size If you want generate graph please enter correct graph size". When the program runs and Graph window is open, user provided incorrect integer number to create graph before, and user pressed generate graph button.

Description

"Incorrect graph size If you want generate graph please enter correct graph size".

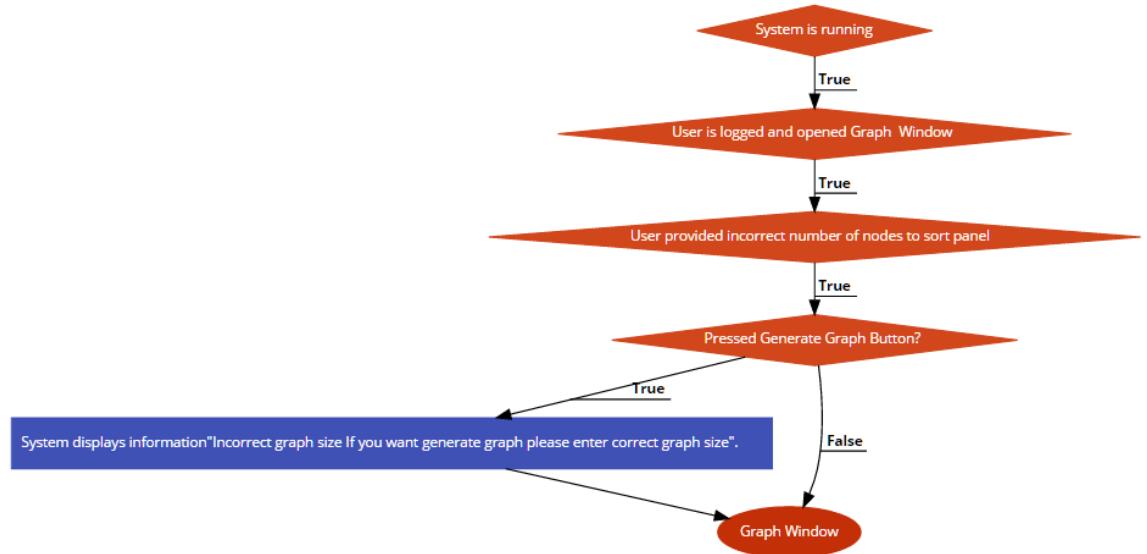


Figure 48: Flow Diagram FR 48

Termination Flow Description

Precondition

System is running, user is logged, Graph Window is open, user provided incorrect integer number to graph label.

Activation

This use case starts when a user presses Generate Graph button after an incorrect number was provided to graph label.

Main flow

- A1. User presses Generate Graph button
- A2. System displays information "Incorrect graph size If you want generate graph please enter correct graph size".

Alternate flow

- B1: Generate Graph button is not pressed.

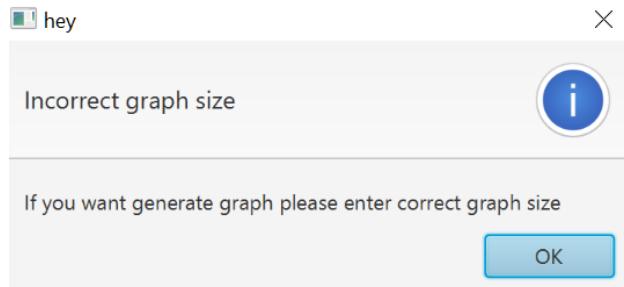
Termination

The user presses Generate Graph button after providing an incorrect number of nodes to graph label.

Post condition

The system displays information "Incorrect graph size If you want generate graph please enter correct graph size".

Post condition Mock



4.1.29 Requirement 55:"FR55"

4.1.29.1 Description & Priority

Display information: please enter correct index for start node and for end node and click Set button, when the provided value of start end node was incorrect. When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button. When the program runs, user is logged in, and Graph window is opened, and user provided incorrect number of nodes to create graph, and press buttonet start/end node in graph.

4.1.29.2 Use Case

Scope

The scope of this use case is Display information "Please enter correct index for start node and for end node and click Set button". When the provided value of start end node was incorrect. When provided value of start, end node in the Graph is incorrect and user pressed set start/end node button.

Description

"Incorrect graph size If you want generate graph please enter correct graph size".

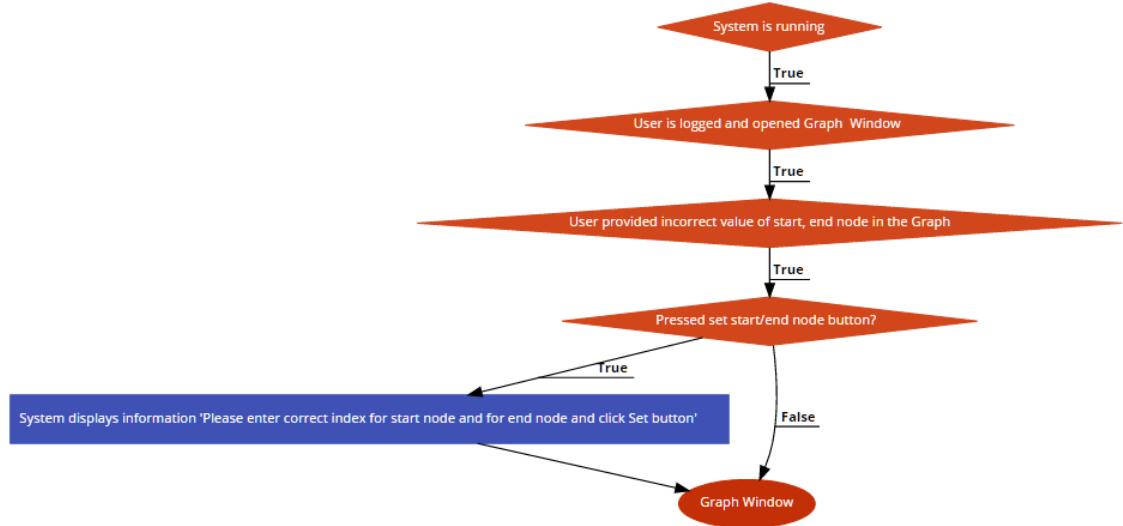


Figure 55: Flow Diagram FR 55

Termination Flow Description

Precondition

System is running, user is logged, Graph Window is open, and user provided incorrect number of nodes to create graph, and press button start/end node in graph.

Activation

This use case starts when a user pressed set start/end node button after provided value of start, end node in the Graph was incorrect.

Main flow

- A1. User press set start/end node button
- A2. System displays information " Incorrect graph size If you want generate graph please enter correct graph size ".

Alternate flow

- B1: start/end node in button in graphs window is not pressed.

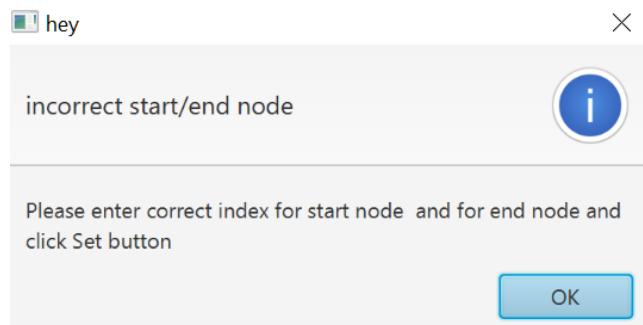
Termination

The user press start/end node button after provided incorrect value of start, end node in the Graph .

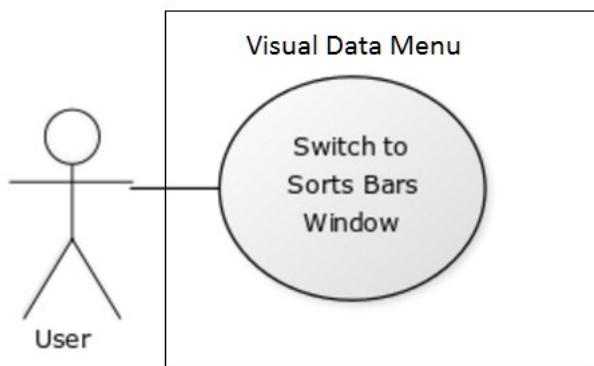
Post condition

The system displays information "Please enter correct index for start node and for end node and click Set button ".

Post condition Mock



4.1.30 Requirement 60:"FR60"



4.1.30.1 Description & Priority

Switch to Sorts Bars window from Menu Window after Sorts Bars button is pressed. When the program runs and Menu window is opened. System should switch to Sorts Bars window after Sorts Bars button is pressed. User is logged in, Menu window is opened, System is running.

4.1.30.2 Use Case

Scope

The scope of this use case is to Switch to Sorts Bars window from Menu window.

Description

This use case describes the process of switching to Sorts Bars window from Menu window.

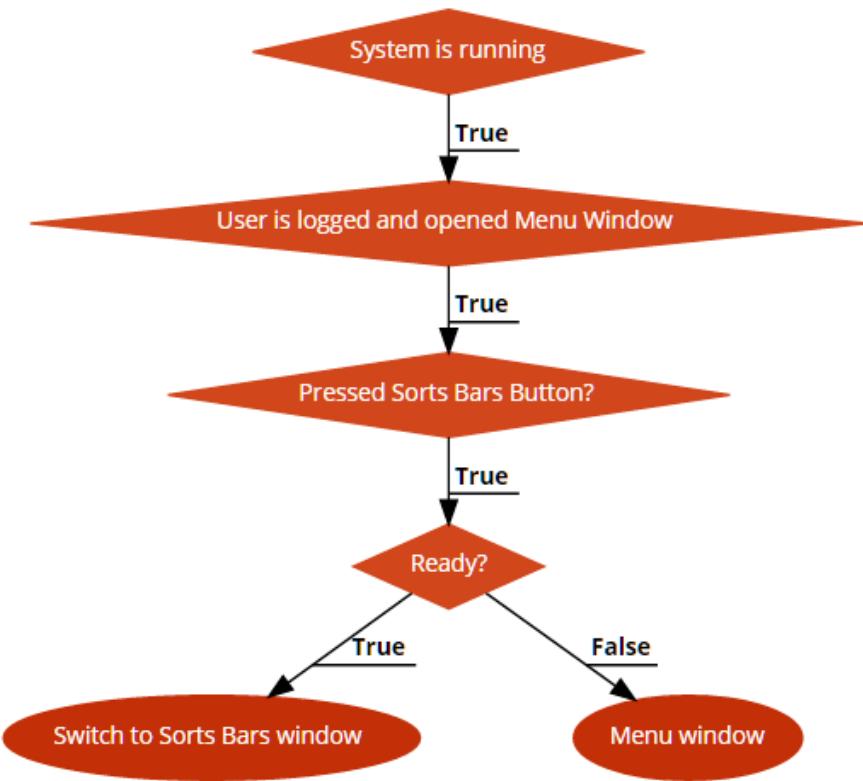


Figure 60 Flow Diagram FR60

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Sorts Bars button.

Main flow

A1. User is in Menu Window and press Sorts Bars button.

A2. System switch to Sorts Bars window

Alternate flow

B1: The system shows Menu window.

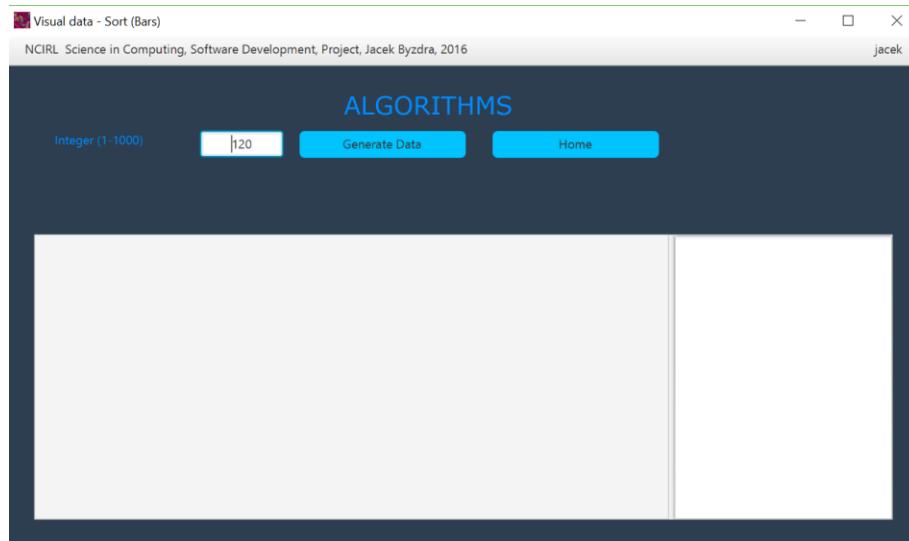
Termination

The user press Sorts Bars button

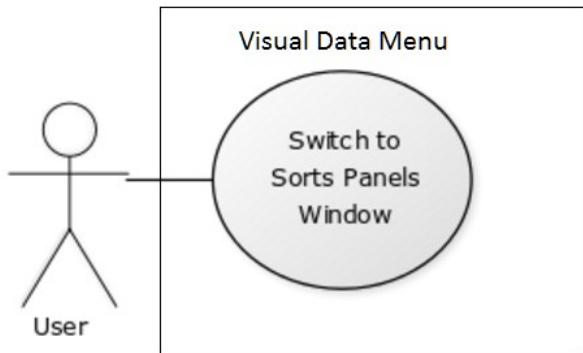
Post condition

The system goes to Sorts Bars window.

Post Mock-up



4.1.31 Requirement 61:"FR61"



4.1.31.1 Description & Priority

Switch to Sorts Panels window from Menu Window after Sorts Panels button is pressed. When the program runs and Menu window is opened. System should switch to Sorts Panels window after Sorts Panels button is pressed. User is logged in, Menu window is opened, System is running.

4.1.31.2 Use Case

Scope

The scope of this use case is to Switch to Sorts Panels window from Menu window.

Description

This use case describes the process of switching to Sorts Panels window from Menu window.

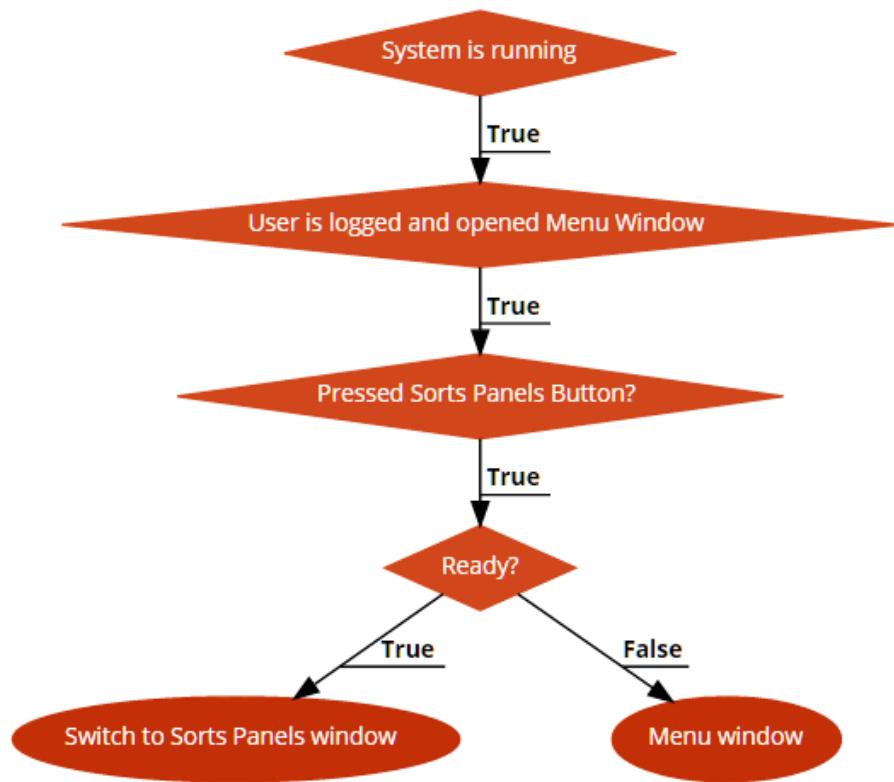


Figure 61 Flow Diagram FR61

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Sorts Panels button.

Main flow

A1. User is in Menu Window and press Sorts Panels button.

A2. System switch to Sorts Panels window

Alternate flow

B1: The system shows Menu window.

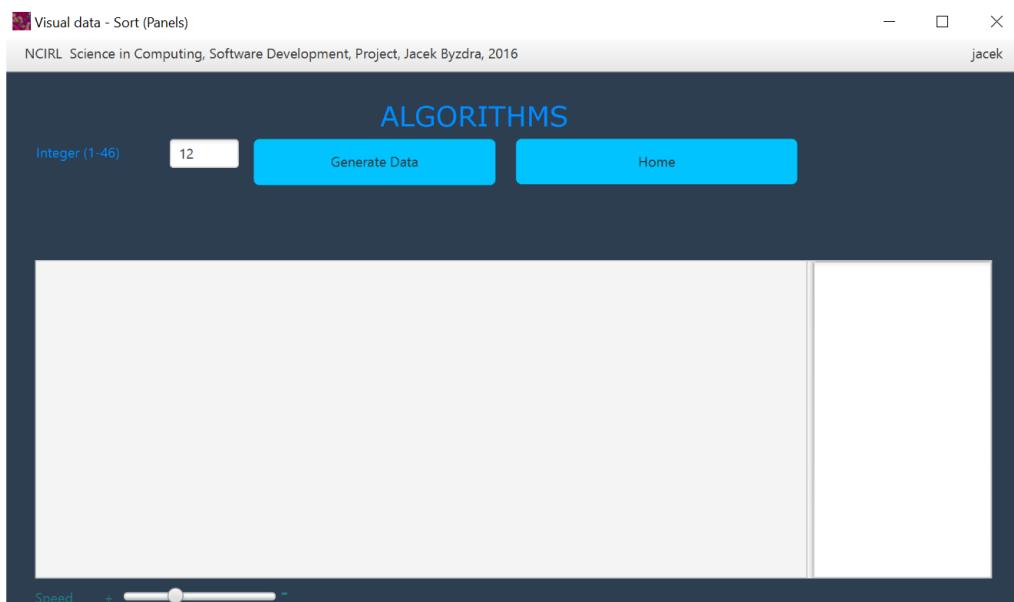
Termination

The user press Sorts Panels button

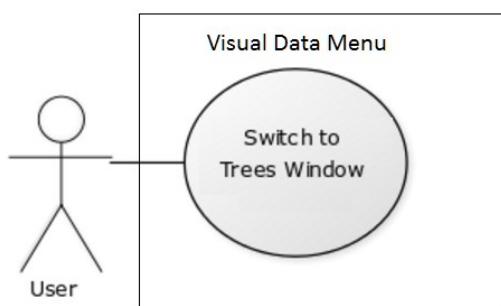
Post condition

The system goes to Sorts Panels window.

Post Mock-up



4.1.32 Requirement 62:"FR62"



4.1.32.1 Description & Priority

Switch to Trees window from Menu Window after Trees button is pressed. When the program runs and Menu window is opened. System should switch to Trees

window after Trees button is pressed. User is logged in, Menu window is opened, System is running.

4.1.32.2 Use Case

Scope

The scope of this use case is to Switch to Trees window from Menu window.

Description

This use case describes the process of switching to Trees window from Menu window.

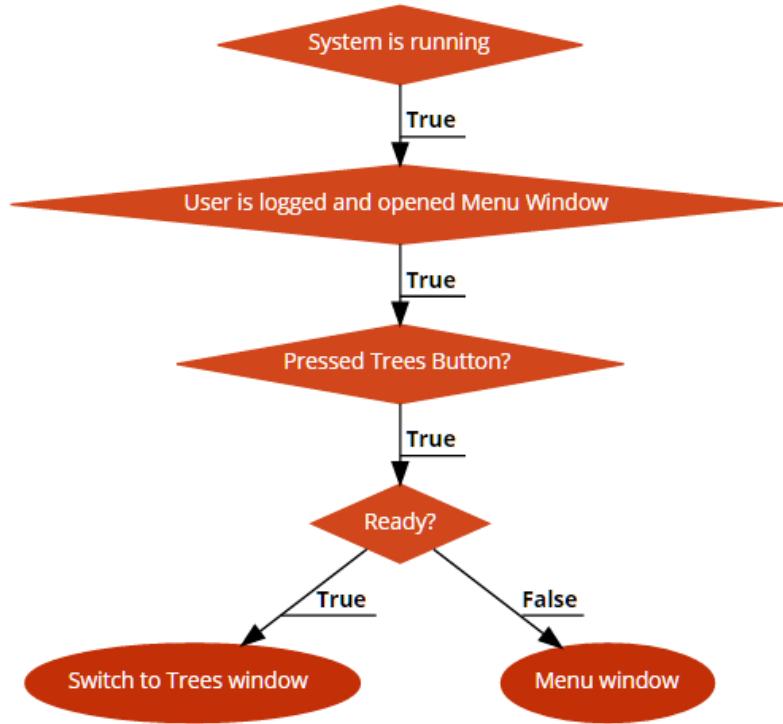


Figure 62 Flow Diagram FR62

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Trees button.

Main flow

A1.User is in Menu Window and press Trees button.

A2. System switch to Trees window

Alternate flow

B1: The system shows Menu window.

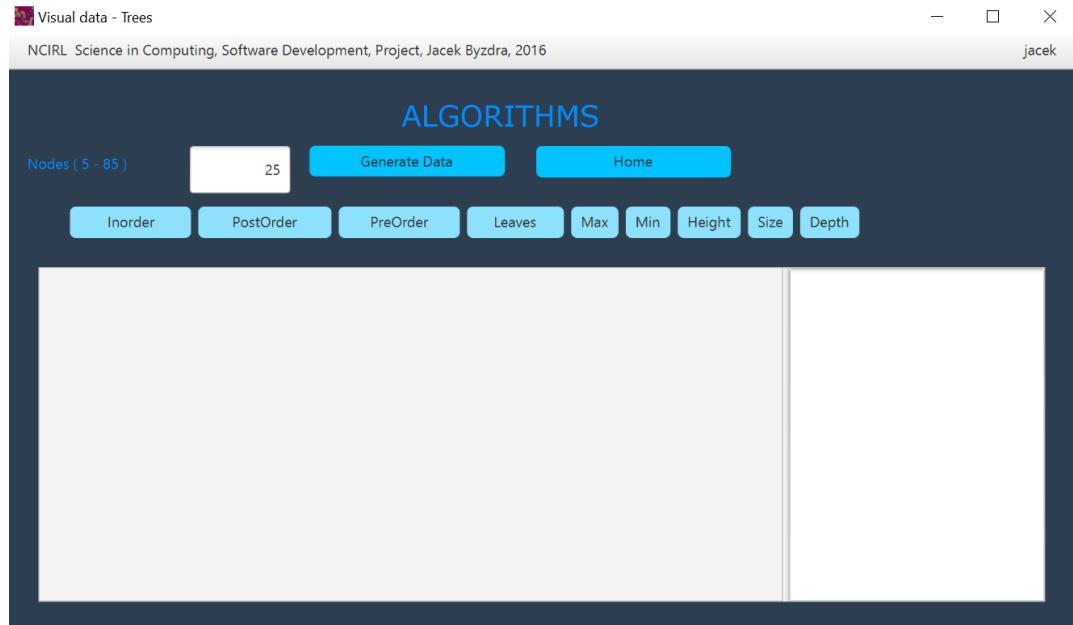
Termination

The user press Trees button

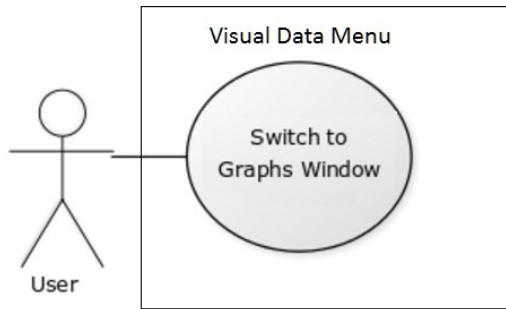
Post condition

The system goes to Trees window.

Post Mock-up



4.1.33 Requirement 63:"FR63"



4.1.33.1 Description & Priority

Switch to Graphs window from Menu Window after Graphs button is pressed. When the program runs and Menu window is opened. System should switch to Graphs window after Graphs button is pressed. User is logged in, Menu window is opened, System is running.

4.1.33.2 Use Case

Scope

The scope of this use case is to Switch to Graphs window from Menu window.

Description

This use case describes the process of switching to Graphs window from Menu window.

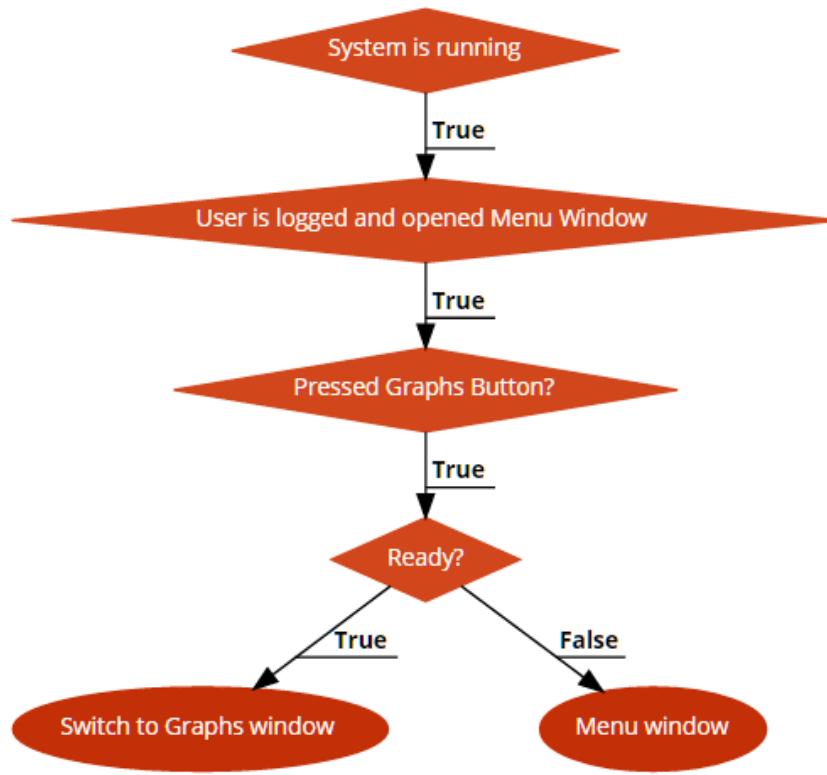


Figure 63 Flow Diagram FR63

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Graphs button.

Main flow

- A1. User is in Menu Window and press Graphs button.
- A2. System switch to Graphs window

Alternate flow

- B1: The system shows Menu window.

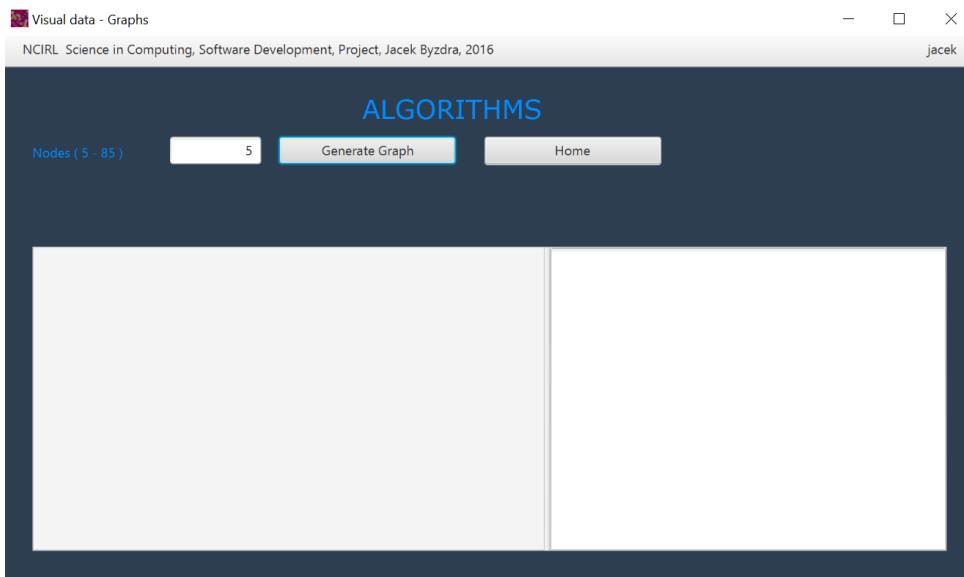
Termination

The user press Graphs button

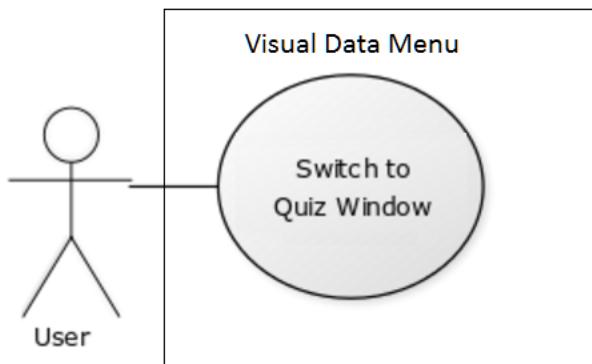
Post condition

The system goes to Graphs window.

Post Mock-up



4.1.34 Requirement 64:"FR64"



4.1.34.1 Description & Priority

Switch to Quiz window from Menu Window after Quiz button is pressed. When the program runs and Menu window is opened. System should switch to Quiz

window after Quiz button is pressed. User is logged in, Menu window is opened, System is running.

4.1.34.2 Use Case

Scope

The scope of this use case is to Switch to Quiz window from Menu window.

Description

This use case describes the process of switching to Quiz window from Menu window.

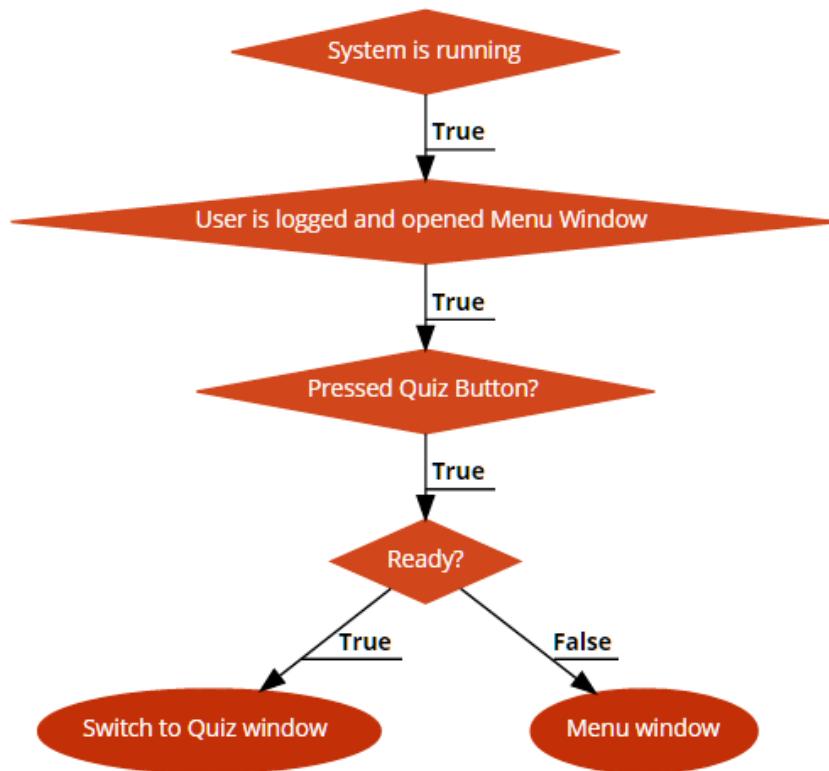


Figure 64 Flow Diagram FR64

Termination Flow Description

Precondition

User is logged in, System is running and Menu Window is open..

Activation

This use case starts when a user press Quiz button.

Main flow

A1. User is in Menu Window and press Quiz button.

A2. System switch to Quiz window

Alternate flow

B1: The system shows Menu window.

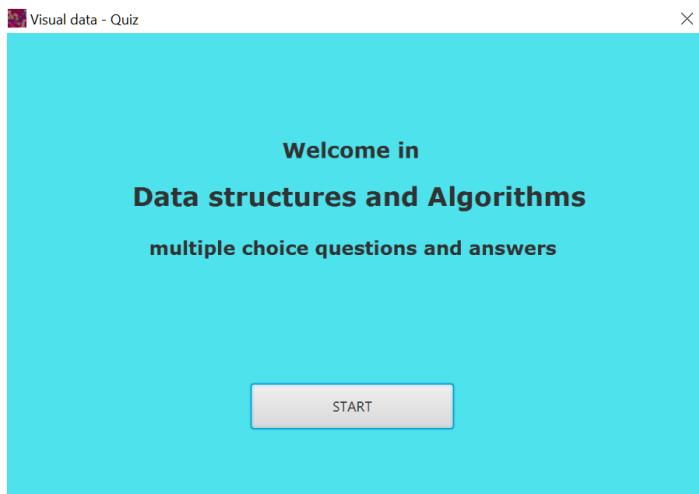
Termination

The user press Quiz button

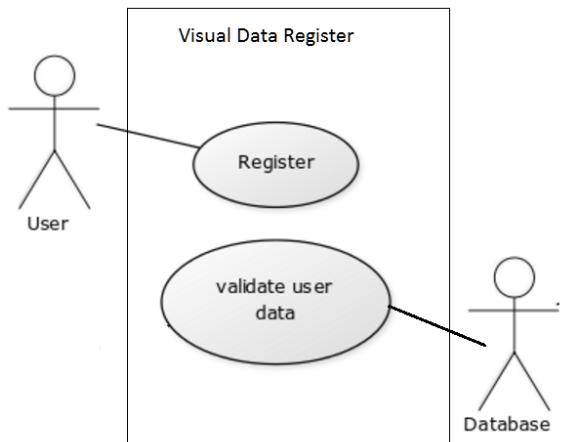
Post condition

The system goes to Quiz window.

Post Mock-up



4.1.35 Requirement 79:"FR79"



4.1.35.1 Description & Priority

Display information: 'User name is incorrect. Please enter correct user name for register'. When the user name was not provided in registration form in Registration Window and Sign Up button was pressed. When the program runs in Register Window and user not provided user name and pressed Sign Up button. Program runs in Register Window.

4.1.35.2 Use Case

Scope

The scope of this use case is Display information 'User name is incorrect. Please enter correct user name for register'. When the user name was not provided in registration form in Registration Window and Sign Up button was pressed. When the program runs in Register Window and user not provided user name and pressed Sign Up button. Program runs in Register Window..

Description

'User name is incorrect. Please enter correct user name for register'.

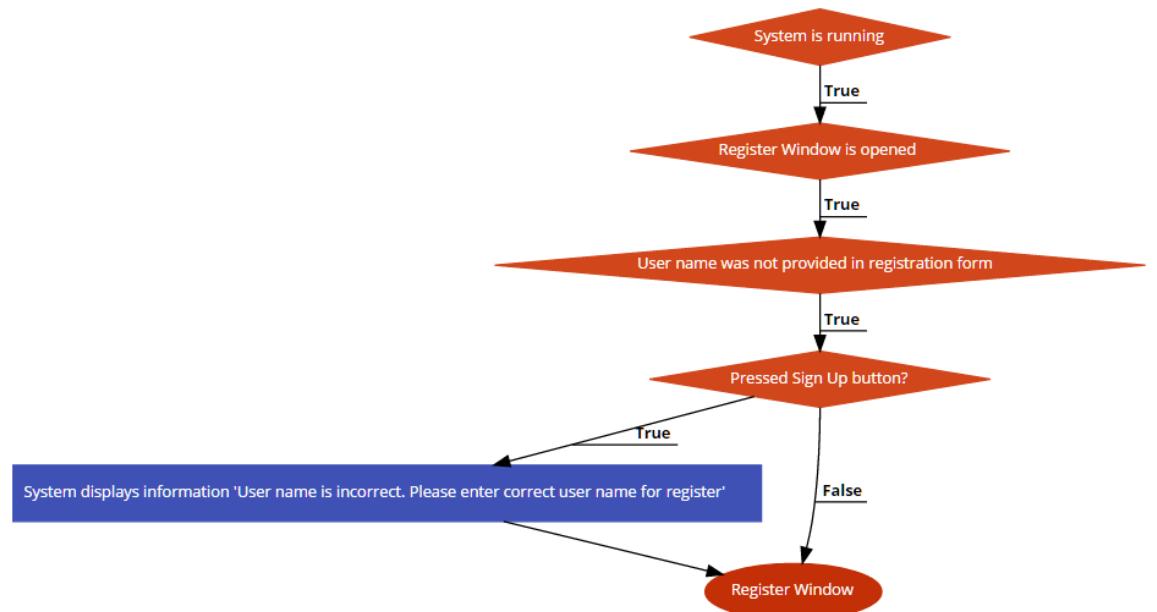


Figure 79: Flow Diagram FR 79

Termination Flow Description

Precondition

System is running, Program runs in Register Window.

Activation

This use case starts when a user pressed Sign Up button after user name was not provided in registration form in Registration Window.

Main flow

- A1. User press Sign Up button
- A2. System displays information "User name is incorrect. Please enter correct user name for register".

Alternate flow

- B1: Sign Up button in Registration Window is not pressed.

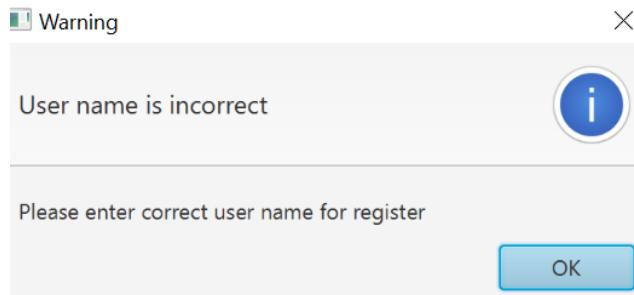
Termination

User not provided user name and pressed Sign Up button.

Post condition

The system displays information ‘User name is incorrect. Please enter correct user name for register’.

Post condition Mock



4.2 Non-Functional Requirements

4.2.1 Performance/Response time requirement

Technologies used in the system allow to perform functions of the system almost instantly.

4.2.2 Availability requirement

System is designed as a tool for students of computer science. The availability of the system depends on distributing channels.

4.2.3 Recover requirement

System is designed to recover automatically.

4.2.4 Robustness requirement

The system does not stop the execution of the program, even if an error occurs.

4.2.5 Security requirement

Security requirements relate to user data and program data stored in a MySQL database.

4.2.6 Reliability requirement

System should work without any faults.

4.2.7 Requirement Maintainability

The requirement maintainability concerns the maintenance of MySQL database.

4.2.8 Reusability requirement

The reusability requirements is related to use the program by students of computer science.

5.4 Project Analysis & Design

The system was designed in Java and JavaFX language to provide an interactive environment for learning. The program allows the user to display on the monitor screen data structures: graphs and trees, sorting algorithms, and interactive quiz, and hints about the data structures and algorithms. The user may perform basic operations on the graphs and trees, change the size of graph, tree, and size of the list generated for searching algorithms. The program visualize and animate the searching algorithms, and the algorithms based on the tree and the graph data structure.

The software architecture of the program was based on the Model View Controller (MVC) pattern.

The view was designed on FXML markup script language. The controller was focused on logic, and model was designed as main model in the system. JavaFX Cascading Style Sheet CSS was used to apply styles to view.

The structure of the files in the system is presented in below diagram.

Visual Data

Models	Views	Controllers	Database
AlgorithmDescription.java BST.java BSTNode.java DBConnection.java Graph.java GraphEdge.java GraphNode.java LoginUser.java QuizQuestion.java UsedNumberList.java WelcomeApp.java	TreeBSTView.fxml admin.fxml description.fxml graph.fxml home.fxml login.fxml menu.fxml quiz.fxml register.fxml sort.fxml welcome.fxml style.css	AdminController.java DescriptionController.java GraphController.java HomeController.java LoginController.java MenuController.java QuizController.java RegisterController.java SortController.java Sort.java TreeBSTViewController.java WelcomeController.java	MySQL database MySQL JDBC driver

Figure: Structure of Visual Data system

The graphic user interface of the project was built entirely with JavaFX running on Java 8. The fxml JavaFX markup language files used for views, comprise hierarchical structure of all GUI elements. The fxml files are user interfaces of JavaFX application. JavaFX Cascading Style Sheet (CSS) was used to provide definitions for style used in the project. The style is used by controls and objects in the scene to change color and fonts. The main application class was created in the file WelcomeApp.java.

The program consists of several MVC sections:

- Section Welcome,
- Section Login,
- Section Register,

- Section Menu,
- Section Admin,
- Section Quiz,
- Section Sorts Bars,
- Section Sorts Panels,
- Section Binary Search Tree,
- Section Graph.

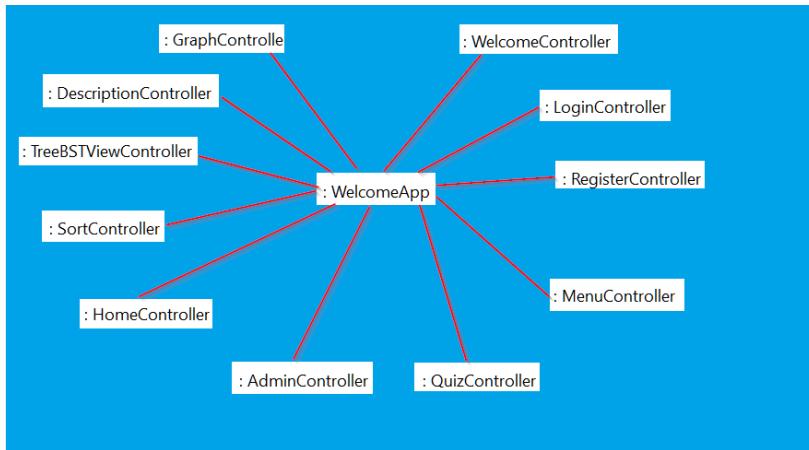


Figure: MVC sections Visual Data

5.4.1 Design and Architecture – Welcome Section

The section Welcome based on the files: WelcomeApp.java, WelcomeController.java, welcome.fxml is initiated immediately after startup of the program.

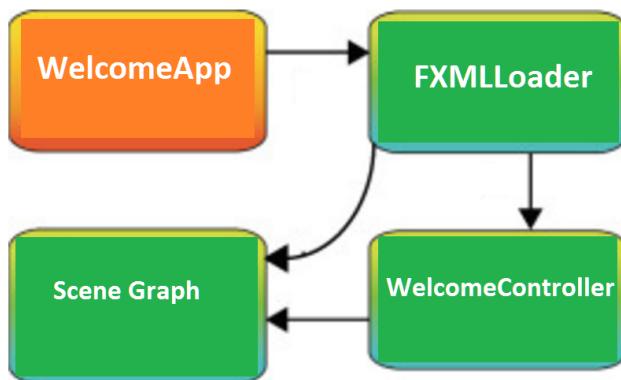


Figure: Structure of JavaFX WelcomeApp Application – Welcome Controller

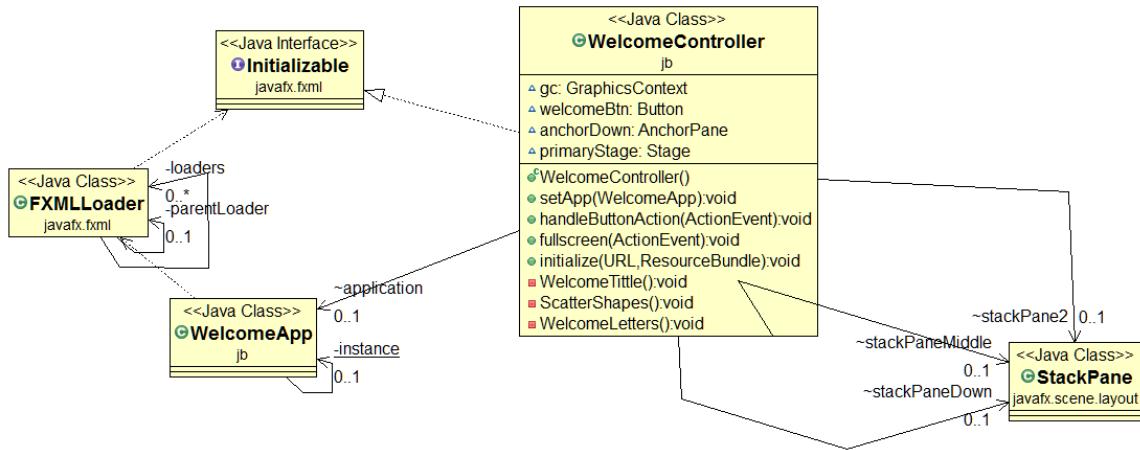


Figure: Welcome Diagram Class

The design of the graph objects and the GUI interface of the program was based on JavaFX FXML markup language. The views were based on separate FXML files.

The structure of FXML used in the project includes a class instance and a properties of class instance: (Introduction to FXML, Release: JavaFX 8.0, www.oracle.com).

The **FXMLLoader** class loads an FXML source file, creates instance of a class, when the tag of fxml file begins with uppercase, and returns the resulting graph elements. A property of a class instance in FXML file is an attribute used to configure the properties of the class.

5.4.2 Design and Architecture – Login Section

The method `handleButtonAction` in **Welcome** view invokes `gotoLogin()` method from **WelcomeApp.java** main class.

The MVC structure of the login section is composed of file:

`LoginUser.java`, `WelcomeApp.java`, `LoginController.java`, `login.fxml`.

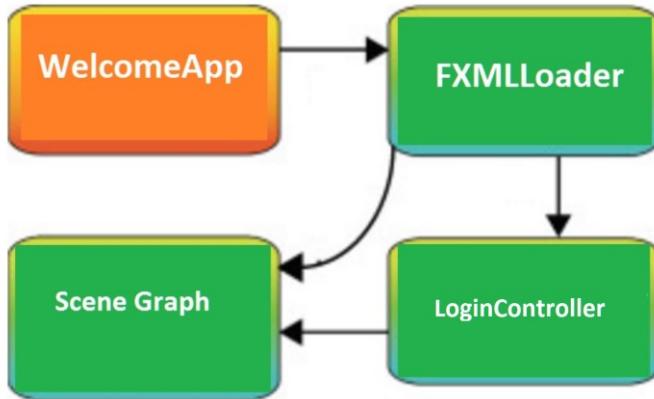


Figure: Structure of JavaFX WelcomeApp Application – Login Controller

In the login section MySQL database is used to validate user name and password.

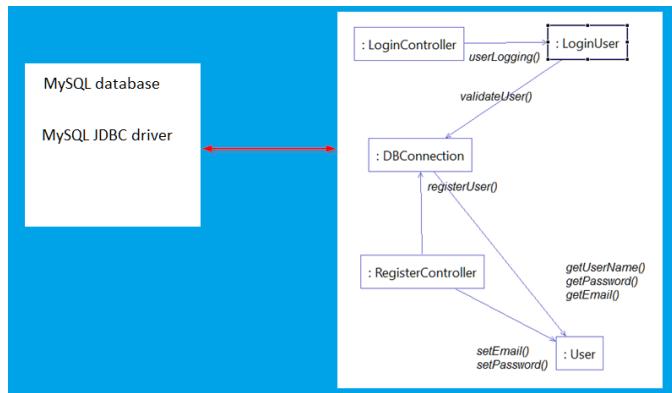


Figure: Login section Visual Data

5.4.3 Design and Architecture – Register Section

The MVC structure of the register section is composed of file:

WelcomeApp.java, RegisterController.java, register.fxml.

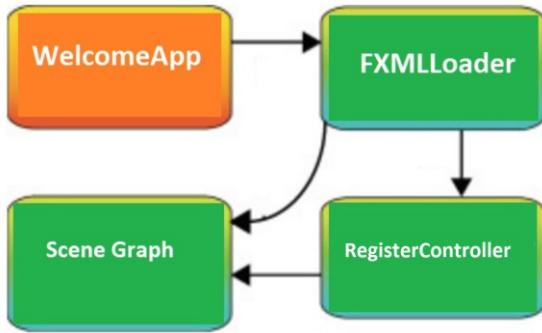


Figure: Structure of JavaFX WelcomeApp Application – Register Controller

5.4.4 Design and Architecture – Menu Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

QuizQuestion.java

MenuController.java

QuizController.java

DescriptionController.java

admin.fxml

description.fxml

menu.fxml

quiz.fxml

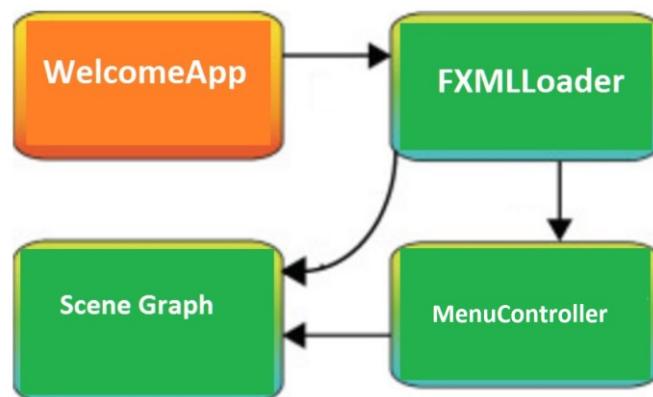


Figure: Structure of JavaFX WelcomeApp Application – Menu Controller

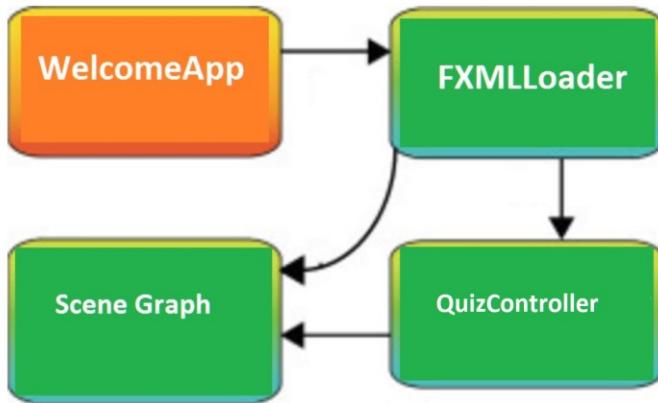


Figure: Structure of JavaFX WelcomeApp Application – QuizController

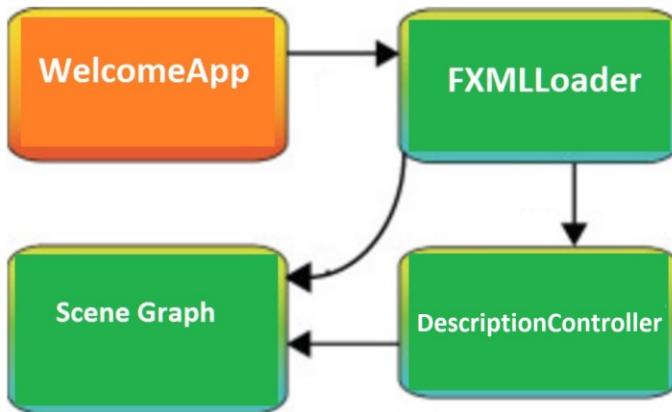


Figure: Structure of JavaFX WelcomeApp Application – Description Controller

5.4.5 Design and Architecture – Bars Sorts Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

HomeController.java

home.fxml

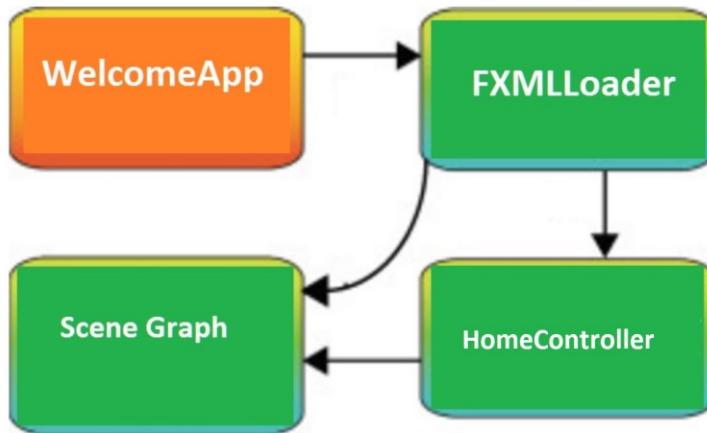


Figure: Structure of JavaFX WelcomeApp Application – Home Controller

5.4.6 Design and Architecture – Panels Sorts Section

The MVC structure of the register section is composed of file:

WelcomeApp.java

Sort.java

SortController.java

sort.fxml

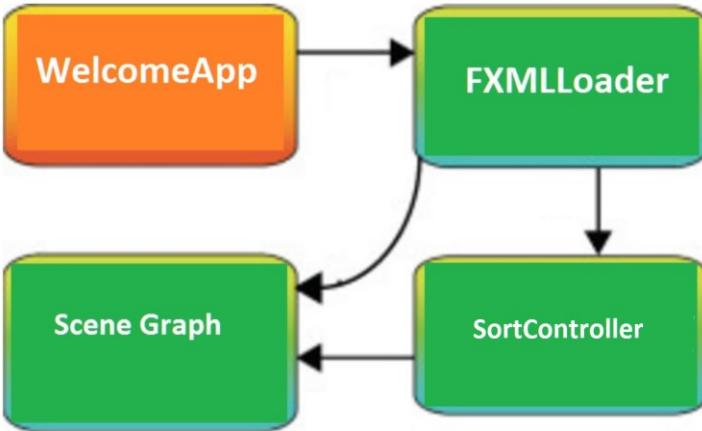


Figure: Structure of JavaFX WelcomeApp Application – Sort Controller

5.4.7 Design and Architecture – Tree Section

The MVC structure of the register section is composed of file:

WelcomeApp.java
TreeBSTView.fxml
TreeBSTViewController.java
BST.java
BSTNode.java

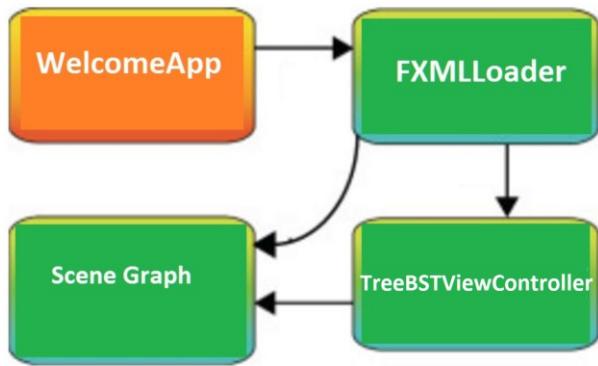


Figure: Structure of JavaFX WelcomeApp Application – Tree Controller

5.4.8 Design and Architecture – Graph Section

The MVC structure of the register section is composed of file:

WelcomeApp.java
Graph.java
GraphController.java
GraphEdge.java
GraphNode.java
graph.fxml

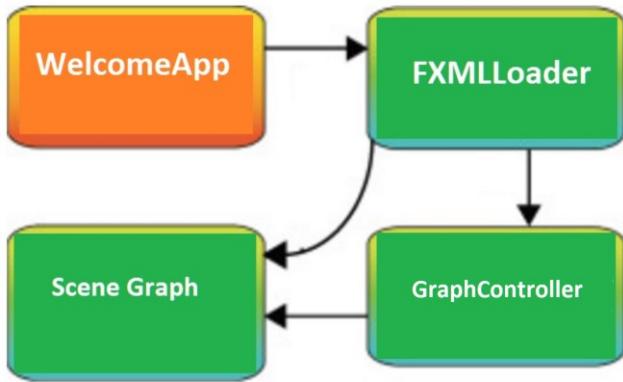


Figure: Structure of JavaFX WelcomeApp Application – Graph Controller

5.5 Project Test Plans

Application test consists of two parts.

The first one concerns on the GUI and checks the functionalities of the program Visual Data (VDJB) to ensure it runs and comply to the requirements. The manual tests allowed to assess whether all the planned requirements of application have been implemented.

Verification of the tests was checked according to the following table functionality and test scenarios:

Test Number	Test Description	Passed	Not Passed
Tst1	Functional requirements of the system	V	
Tst2	Access to Visual Data System	V	
Tst3	Start and open program in different screen size mode	V	
Tst4	Log in user	V	
Tst5	Log out user	V	
Tst6	Register user	V	
Tst7	Access to Welcome view	V	
Tst8	Performance of Welcome view	V	
Tst9	Access to Sorts Bars view	V	
Tst10	Performance of Sorts Bars view	V	
Tst11	Algorithms descriptions in Sorts Bars view	V	
Tst12	Enter numbers of elements to sort and generate data in Sorts Bars view	V	

Tst13	Bubble sort method presentation in Sorts Bars view	V	
Tst14	Insertion sort method presentation in Sorts Bars view	V	
Tst15	Selection sort method presentation in Sorts Bars view	V	
Tst16	Merge sort method prestenattion in Sorts Bras view	V	
Tst17	Quick sort method presentation in Sorts Bars view	V	
Tst18	Bubble sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst19	Insertion sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst20	Selection sort method presentation with maximum number of elements in Sorts Bars view	V	
Tst21	Merge sort method prestenattion in with maximum number of elements Sorts Bras view	V	
Tst22	Quick sort method presentation in with maximum number of elements Sorts Bars view	V	
Tst23	Bubble sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst24	Insertion sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst25	Selection sort method presentation with minimum number of elements in Sorts Bars view	V	
Tst26	Merge sort method prestenattion in with minimum number of elements Sorts Bras view	V	
Tst27	Quick sort method presentation in with minimum number of elements Sorts Bars view	V	
Tst28	Reliability of the system when all buttons are pressed when program is running	V	
Tst29	Time of sorting visibility for each sort method in Sorts Bars view	V	
Tst30	Switch back to Menu option from Sorts Bars view	V	
Tst31	Access to Sorts Panels view	V	
Tst32	Performance of Sorts Panels view	V	
Tst33	Enter numbers of elements to sort and generate data in Sorts Panels view	V	
Tst34	Bubble sort method presentation in Sorts Panels view	V	
Tst35	Insertion sort method presentation in Sorts Panels view	V	
Tst36	Selection sort method presentation in Sorts Panels view	V	
Tst37	Heap sort method prestenattion in Sorts Panels view	V	
Tst38	Quick sort method presentation in Sorts Panels view	V	
Tst39	Slide speed change of animation panels	V	

Tst40	Low speed Bubble sort method presentation in Sorts Panels view	V	
Tst41	Low speed Insertion sort method presentation in Sorts Panels view	V	
Tst42	Low speed Selection sort method presentation in Sorts Panels view	V	
Tst43	Low speed Heap sort method prestenattion in Sorts Panels view	V	
Tst44	Low Quick sort method presentation in Sorts Panels view	V	
Tst45	High speed Bubble sort method presentation in Sorts Panels view	V	
Tst46	High speed Insertion sort method presentation in Sorts Panels view	V	
Tst47	High speed Selection sort method presentation in Sorts Panels view	V	
Tst48	High speed Heap sort method prestenattion in Sorts Panels view	V	
Tst49	High Quick sort method presentation in Sorts Panels view	V	
Tst50	Low speed Bubble sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst51	Low speed Insertion sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst52	Low speed Selection sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst53	Low speed Heap sort method prestenattion in with maximum number of elements Sorts Panels view	V	
Tst54	Low speed Quick sort method presentation in with maximum number of elements Sorts Panels view	V	
Tst55	High speed Bubble sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst56	High speed Insertion sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst57	High speed Selection sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst58	High speed Heap sort method prestenattion in with minimum number of elements Sorts Panels view	V	
Tst59	High speed Quick sort method presentation in with minimum number ofelements Sorts Panels view	V	

Tst60	Low speed Bubble sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst61	Low speed Insertion sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst62	Selection sort method presentation with maximum number of elements in Sorts Panels view	V	
Tst63	Low speed Heap sort method prestenattion in with maximum number of elements Sorts Panels view	V	
Tst64	Low speed Quick sort method presentation in with maximum number of elements Sorts Panels view	V	
Tst65	Low speed Bubble sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst66	Low speed Insertion sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst67	Low speed Selection sort method presentation with minimum number of elements in Sorts Panels view	V	
Tst68	Low speed Heap sort method prestenattion in with minimum number of elements Sorts Panels view	V	
Tst69	Low speed Quick sort method presentation in with minimum number of elements Sorts Panels view	V	
Tst70	Switch back to Menu option from Sorts Panels view	V	
Tst71	Access to Trees view	V	
Tst72	Performance of Trees view	V	
Tst73	Enter numbers of elements to generate BST Tree in Trees view	V	
Tst74	Inorder traversal option with path presentation in Trees view	V	
Tst75	Postorder traversal option with path presentation in Trees view	V	
Tst76	Preorder traversal option with path presentation in Trees view	V	
Tst77	Leaves selected in Trees view	V	
Tst78	Find minimum value in Trees view	V	
Tst79	Find maximum value in Trees view	V	
Tst80	Height of tree presentation in Trees view	V	
Tst81	Size of tree presentation in Trees view	V	
Tst82	Depth of tree presentation in Trees viw	V	
Tst83	Switch back to Menu option from Trees view	V	
Tst84	Access to user management for logged user	V	
Tst85	Delete user when user is logged	V	
Tst86	Set Admin role for user	V	

Tst87	Access to quiz questions management	V	
Tst88	Add new quiz question	V	
Tst89	Update quiz question and its answers	V	
Tst90	Delete quiz question	V	
Tst91	Access to Graphs view	V	
Tst92	Performance of Graphs view	V	
Tst93	Enter numbers of elements to generate Graph in Graphs view	V	
Tst94	Performance of search methods in Graphs view	V	

Table: Manual tests Visual Data System

Second part of the testing included automatic testing supported by JUnit - framework for Java programming language. JUnit 4 is integrated in NetBeans IDE 8.1. The NetBeans IDE 8.1 was used for creating and testing application. JUnit unit testing framework enables quickly and easily create test suites.

It increases the quality of code and software reliability. I choosed a few critical classes to build unit tests in this method, especially all the classes responsible for the Searching Sorts and Binary Search Tree algorithms. Each test class corresponds to the application class. The test class consists of test Initializers and Finalizers, and methods for unit cases. Initialization method runs in the tests class before each test case, test finalizer method is running after each test case in the test class. The Junit tests are very helpful when there is need to initialize some variables before run test or to clean up any data when finish.

In JUnit4 each test method should have an @Test annotation if we want it to take part in the running tests.^{1,2}

JUnit tests examples:

Class BSTTest

Initiate BST the established values: 8, 3, 10, 1, 6, 14, 4, 7, 13 to build tree:

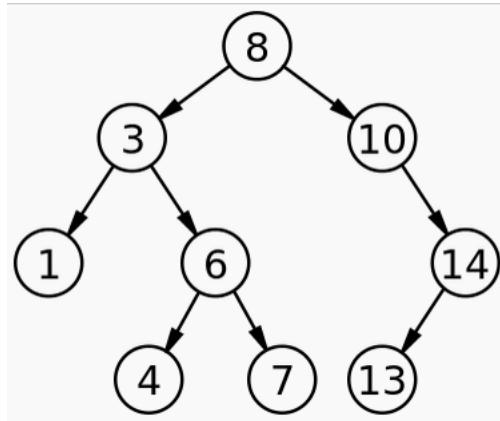


Figure: BST tree

Methods:

setUp() - to init BST before unit case

testGetRoot() - method return a root - should return 8

testIsEmpty() - method check if tree is empty - should return false

testSearch() - method search data in BST, for value 6 - should return true, for value 5 - false;

testSize() - method return numbers of elements in BST, should return 9

testMaxDepth() - method return depth of BST, the longest path from root to leaf, should return 4

testMinValue() - should return 1

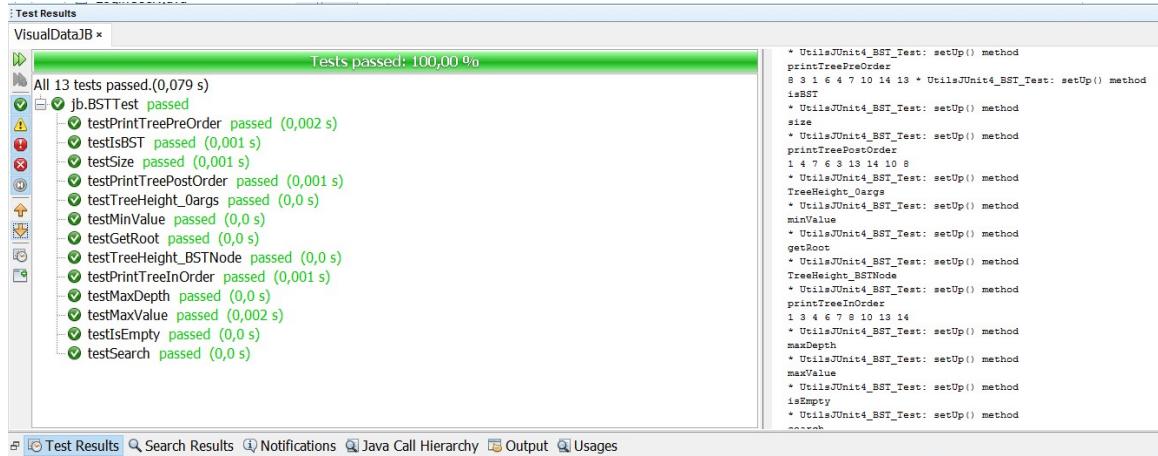
testMaxValue() - should return 14

testPrintTreeInOrder() - method traverse BST in In-Order method, should return path: 1, 3, 4, 6, 7, 8, 10, 13, 14

`testPrintTreePreOrder()` - method traverse BST in Pre-Order method, should return path: 8, 3, 1, 6, 4, 7, 10, 14, 13

`testPrintTreePostOrder()` - method traverse BST in Post- Order method, should return path: 1, 4, 7, 6, 3, 13, 14, 10, 8

Execution of the test shown in the figure:



Class SortTest

In this class I build an init array which fixed values, it helps test my sort method.

Methods:

`setUp()` - to init an array of 12 values before unit case:

```
int[] tab = {8, 3, 10, 1, 6, 14, 4, 7, 13, 2, 19, 5};
```

This array is used to build data list to sort in test methods.

`testBubbleSort()` – sorts data list using algorithm Bubble sort, returns sorted list

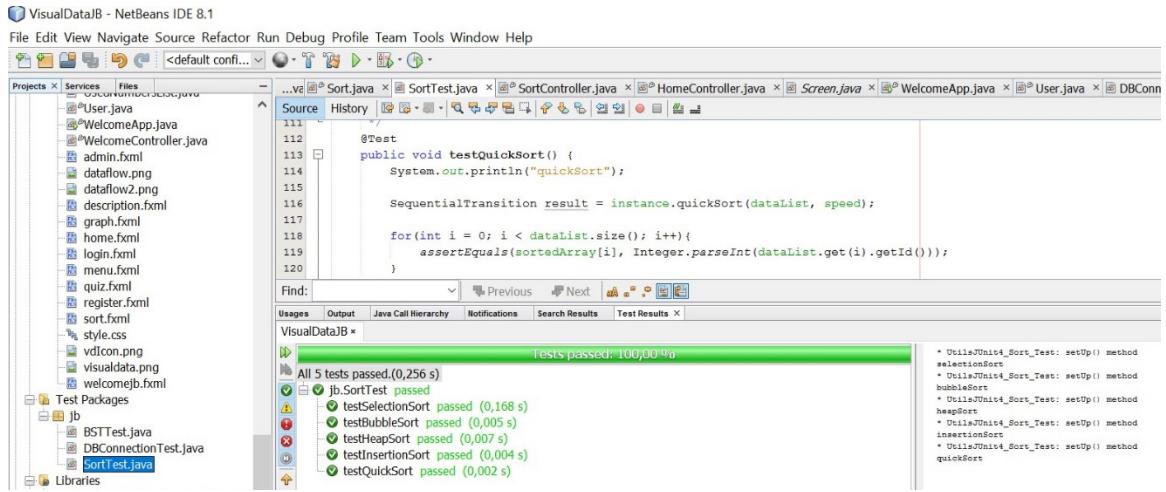
`testInsertionSort()` – sorts data list using algorithm Insertion sort, returns sorted list

`testSelectionSort()` – sorts data list using algorithm Selection sort, returns sorted list

`testQuickSort()` – sorts data list using algorithm Quick sort, returns sorted list

`testHeapSort()` - sorts data list using algorithm Heap sort, returns sorted list

Execution of this test shown in the figure:



Class DBConnectionTest

In this class I test a connection with data base and methods which interact with database.

This class doesn't init any variables in `setUp()` method.

Tested method:

`testConnect()` - method checks if connection is correctly created

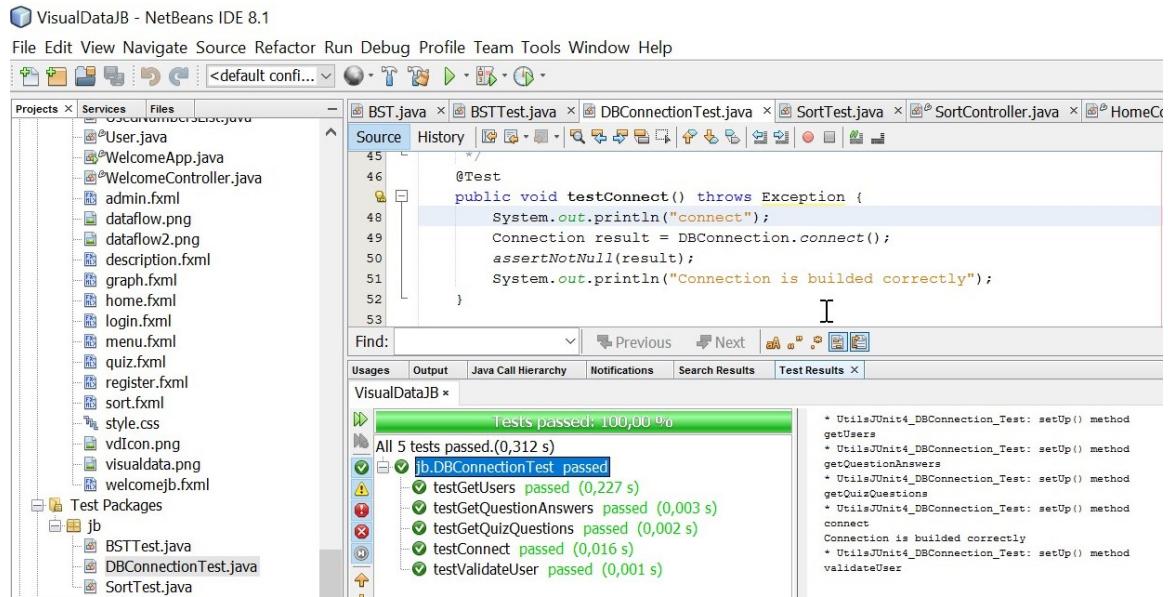
`testValidateUser()` – method validate if user is registered in database

`testGetUsers()` – method returns users form database, count of users should be > 0

`testGetQuizQuestions()` - method returns quiz questions from database, count of questions should be > 0

`testGetQuestionAnswers()` – method tests if question has exactly 4 answers

Execution of this test shown in the figure:



5.6 *Monthly Journals*

5.6.1 Monthly Journal #1

Reflective Journal

Student name: JACEK BYZDRA

Student no: x15030491

Student email: x15030491@student.ncirl.ie

Programme : Higher Diploma in Science in Computing

Specialisation: Software Development

For Month: October

Date: November 06th 2016

6 My Achievements

This month, I was able to do:

1. Analyse requirements of the software in the project and , and implement JavaFX platform in the application code. I was able to implement Model View Controller based on FXML views. To provide application platform independent I chose Java/JavaFX software in this project.
2. Develop and implement most part of the core application layer in the project. As the core layer I provided following solution:
 - a) Interactive sorting algorithms: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort.
 - b) Data structures: Array, Array List, Binary Search Tree,
 - c) Methods for the data structures creation, and searching.
 - d) Methods for visual presentation of the algorithms, and data structures.
3. Develop database layer based on MySQL database,
4. Develop User Access to application, and database.
5. Tests of above algorithms, and data structures.
6. Development of Graphical User Interface based on Model View Controller: FXML views, FXML controllers, and Models created in JavaFX/Java language.

My contributions to the projects included:

1. All points mentioned in previous topic , and more: Create visualization application based on 2D/3D shapes.

7 My Reflection

The application written in JavaFX/Java language seems to be optimal solution for the project.

I had to battle with some problems:

1. The model view controller written in JavaFX could be based on two solutions: FXML scene based on XML interface, or scene built in Java application. The scene is based on nodes built in hierarchy , where each level of the nodes may have own classes. I chose in the beginning nodes in which I couldn't create all required graphic shapes. Finally after some additional tests I chose proper interface.

Part of the solution example is place below

```
<AnchorPane fx:id="window" minHeight="-Infinity" minWidth="-Infinity" prefHeight="495.0"
    prefWidth="885.0" stylesheets="@style.css" xmlns="http://javafx.com/javafx/8"
    xmlns:fx="http://javafx.com/fxml/1" fx:controller="jb.WelcomeController">
    <children>
        <HBox AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="-1.0">
            <children>
                <MenuBar minWidth="-Infinity" prefHeight="29.0" prefWidth="335.0"
                    HBox.hgrow="ALWAYS">
                    <menus>
                        <Menu mnemonicParsing="false" text="Transition Animation">
                            <items>
                                <MenuItem id="fullscreen" fx:id="fullscreen" mnemonicParsing="false"
                                    onAction="#fullscreen" text="Fullscreen" />
                                <MenuItem id="transition1" fx:id="transition1" mnemonicParsing="false"
                                    text="transition1" />
                                <MenuItem id="menuPath2" fx:id="menuPath2" mnemonicParsing="false"
                                    text="Path2" />
                            </items>
                        </Menu>
                    </menus>
                </MenuBar>
                <MenuBar fx:id="userMenuBar" layoutX="719.0" maxWidth="190.0" minWidth="-Infinity"
                    prefHeight="29.0" HBox.hgrow="SOMETIMES">
                    <menus>
                        <Menu fx:id="userMenu" mnemonicParsing="false" text="User">
                            <items>
                                <MenuItem fx:id="OptionsButton" mnemonicParsing="false" text="Register" />
                                <MenuItem fx:id="logoutButton" mnemonicParsing="false" text="Login" />
                            </items>
                        </Menu>
                    </menus>
                </MenuBar>
            </children>
        </HBox>
    </children>
</AnchorPane>
```

```

public class WelcomeController extends BorderPane implements Initializable {

    GraphicsContext gc;

    @FXML
    Button welcomeBtn;

    @FXML
    VBox vbox1;

    @FXML
    Canvas canvas1;

    @FXML
    MenuItem transition1;

    @FXML
    MenuItem menuPath2;

    @FXML
    StackPane stackPane1;

    @FXML
    StackPane stackPane2;

    Stage primaryStage;
    WelcomeApp application;

    public void setApp(WelcomeApp application){
        this.application = application;
    }

    @FXML
    public void handleButtonAction(ActionEvent event) throws IOException{
        if(event.getSource() == welcomeBtn){
            WelcomeApp.getInstance().gotoLogin();
        }
    }

    @FXML
    public void fullscreen(ActionEvent event)
    {
        stage.setFullScreen(true);
    }
}

```

```

@Override
public void initialize(URL location, ResourceBundle resources) {
    WelcomeLetters();
    WelcomeTittle();
}

public void WelcomeTittle()
{
    stackPane2.getChildren().clear();

    final Text txt = new Text(10, 50,"Welcome in Visual Data");
    txt.setFont(Font.font ("Verdana", 48));
    txt.setFill(Color.RED);
    stackPane2.getChildren().add(txt);

    StrokeTransition st = new StrokeTransition();
    st.setDuration(Duration.seconds(20));
    //st.setDelay(Duration.seconds(.5));
    st.setShape(txt);
    st.setFromValue(Color.RED);
    st.setToValue(Color.BLUE);
    st.setCycleCount(Timeline.INDEFINITE);
    st.setAutoReverse(true);
    st.setRate(10);
    st.play();
}

public void WelcomeLetters()
{
    stackPane1.getChildren().clear();

    final Text txt1 = new Text(10, 50,"W");
    final Text txt2 = new Text(10, 50,"E");
    final Text txt3 = new Text(10, 50,"L");
    final Text txt4 = new Text(10, 50,"C");
    final Text txt5 = new Text(10, 50,"O");
    final Text txt6 = new Text(10, 50,"M");
    final Text txt7 = new Text(10, 50,"E");

    txt1.setFont(Font.font ("Verdana", 48));
    txt2.setFont(Font.font ("Verdana", 48));
    txt3.setFont(Font.font ("Verdana", 48));
    txt4.setFont(Font.font ("Verdana", 48));
    txt5.setFont(Font.font ("Verdana", 48));
    txt6.setFont(Font.font ("Verdana", 48));
}

```

```

txt7.setFont(Font.font ("Verdana", 48));

        txt1.setFill(Color.RED);
        txt2.setFill(Color.AQUAMARINE);
        txt3.setFill(Color.BLUEVIOLET);
        txt4.setFill(Color.CHARTREUSE);
        txt5.setFill(Color.CORAL);
        txt6.setFill(Color.DARKGREEN);
        txt7.setFill(Color.GOLD);

final Path path = new Path();
path.getElements().add(new MoveTo(340, 340));
path.getElements().add(new ArcTo(20, 20, 5, 160, 60, true, false));
path.getElements().add(new ArcTo(20, 20, 5, 360, 220, true, false));
path.getElements().add(new ArcTo(20, 20, 5, 380, 200, true, false));
path.getElements().add(new ArcTo(20, 20, 5, 60, 240, true, false));
path.setOpacity(0.0);

stackPane1.getChildren().add(path);
stackPane1.getChildren().add(txt1);
final PathTransition pathTransition = new PathTransition();

pathTransition.setDuration(Duration.seconds(8.0));
pathTransition.setDelay(Duration.seconds(.5));
pathTransition.setRate(0.3);
pathTransition.setPath(path);
pathTransition.setNode(txt1);
pathTransition
.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
pathTransition.setCycleCount(Timeline.INDEFINITE);
pathTransition.setAutoReverse(true);
pathTransition.play();

```

2. During implementation of the visualisation method I met the problem with refreshment of text and shapes in the movement methods. I solved the problem implementing transition Java's methods. Part of example is presented below:

```

private ParallelTransition swapElements(StackPane l1, StackPane l2, ArrayList<StackPane> list,
double speed) {
    TranslateTransition t1 = new TranslateTransition();
    TranslateTransition t2 = new TranslateTransition();
    t1.setDuration(Duration.millis(speed));
    t2.setDuration(Duration.millis(speed));
    ParallelTransition pl = new ParallelTransition();
    t1.setNode(l1);

```

```
        t2.setNode(l2);
        t1.setByX(szer+space);
        t2.setByX(-1*(szer+space));
        pl.getChildren().addAll(t1, t2);
        Collections.swap(list, list.indexOf(l1), list.indexOf(l2));
        return pl;
    }
}
```

3. During planning of the project I met the problem: some classes used in JavaFX 2 were deprecated in JavaFX 8 . The problem was recognized by Oracle community (<https://community.oracle.com/thread/2544323>) 'there was mistake made : implementation of Builder class has some intractable problems with respect to binary compatibility . . . this was accomplished using generics, and as it turns out, depended on two bugs in JDK 6 and JDK 7 in order to work . Those bugs were fixed in JDK 8, and as a result, the builders no longer work correctly for certain cases.' Of course I decided to not used classes which were deprecated(listed below).

```
javafx.scene.NodeBuilder<B>
javafx.scene.ParentBuilder<B>
javafx.scene.control.ControlBuilder<B>
javafx.scene.control.LabeledBuilder<B>
javafx.scene.control.ButtonBaseBuilder<B>
javafx.scene.control.ButtonBuilder<B>
```

8 Intended Changes

Next month, I will end the project.

But I realised that I still need to do:

1. Implement Graphs data structures in the application, do some little change of graphic layouts in app, and make some descriptions/help methods for user.
2. Development User Interaction methods,
3. Testing of all application,
4. Make documentation,
5. Deploy app,
6. Do post implementation review : get feedback about product from app user.

9 Supervisor Meetings

First meeting : October 18th 2016.

Action taken: checkout of : requirements of software specification.

Action which should be taken until next meeting November 22nd : review of product, review of post implementation, review of documentation.

JACEK BYZDRA

November 06th , 2016

9.1.1 Monthly Journal #2

Reflective Journal

Student name: JACEK BYZDRA

Student no: x15030491

Student email: x15030491@student.ncirl.ie

Programme : Higher Diploma in Science in Computing

Specialisation: Software Development

For Month: November

Date: November 20th 2016

10 My Achievements

In the time stamps between November 06th to November 20th , I was able to do:

7. Implement Graphs data structures visualisation.
8. The graphic layout of application was improved: the elements in the views (FXML) were aligned with CSS script, not needed elements were removed, the views structure was optimize to windows size,
9. Develop and implement Dijkstra algorithm visualization on implemented Graphs:
 - e) Develop Dijkstra search method to find the shortest paths between start and end nodes in the Graph,
 - f) Develop method which allows user to choose end and start node in the Graph,
 - g) Develop method which allows user to provide number of nodes to be generated in the Graph,
 - h) Develop methods to generate and visualize: Graph,
 - i) Develop Dijkstra method to find and visualize the shortest paths between and end node in the graph,
10. Develop user interaction methods: provide methods for changing algorithms, change the input size of the data.
11. Test of all algorithms manually.
12. The current app was passed for testing one of the student in NCIRL. First feedback is positive.

My contributions to the projects included:

2. All points mentioned in previous topics , and more: The developed Graph visualization shows all nodes in the symetric segments, what improves visual view of the Graph.

11 My Reflection

Up to now the planned application was developed according to the plan. I met below problem which I had to overcome :

4. The dynamic drawing of Java FX shape elements was visible only in the middle of StackPanes. The probem was solved by changing StackPanes to AnchorPanes.

12 Intended Changes

Next month, I will end the project.

I still need to do:

7. Testing of all application,
8. Make documentation ,
9. Deploy app (The first pilot was deployed for testing. The test methods will be implemented in final app),
10. Do post implementation review : get feedback about product from app user (it was partly received),

13 Supervisor Meetings

First meeting : October 18th 2016.

Action taken: checkout of requirements of software specification.

Second meeting: November 20th 2016

Action taken: review of product, review of pilot post implementation.

Action which should be taken until next meeting November 30th : review of final product, review of post implementation, review of documentation, get positive feedback from test user, and from stakeholders.

JACEK BYZDRA

November 20^h , 2016