Presenters of the project:

Amit Furman, 207765108, amitff103@gmail.com

Eden Ismah-Moshe, 209524628, [edenis@mta.ac.il](mailto:edenis@mta.ac.il)

We implemented the third bonus feature, which involved adding charts to the Statistics tab. In consultation with Aviad, we remove the table that was previously present in this tab and replace it entirely with visually appealing and informative charts.

\*Pay attention:

1. we added tooltips to free inputs text fields in flow execution tab.

2. In the Statistics tab, we have implemented tooltips for each bar displayed in the charts. When hovering over a bar, a tooltip is triggered, presenting the name of the bar and its corresponding value.

Our project is a workflow\pipeline system that enables assembling different scenarios (called flows) from common components (called steps), including executing them and producing required results.

It consists of two main modules - JavaFX and systemEngine. The JavaFX module provides a desktop interface for users to interact with the system, while the systemEngine module handles the workflow processing and execution.

The JavaFX module is responsible for communicating with the user and transferring the necessary data to the systemEngine module. It provides a desktop interface for users to interact with the system and performs input validation and error handling to ensure the user's input is correct.

The systemEngine module is responsible for handling workflow processing and execution. It contains several packages, including datadefinition, dto, exceptions, flow, jaxb.schema, statistic, and steps. These packages provide various functionalities, such as defining workflow data structures, carrying data between modules, handling exceptions, defining workflow logic, and storing workflow statistics.

Packages in the systemEngine module:

* The datadefinition package provides implementations of all data definition types used in the system.
* The dto package contains classes that act as containers for information and carry data between the UI and the systemEngine. These classes are used to pass information about workflows, steps, and data definitions between the two modules.
* The exceptions package provides classes for handling errors and exceptions in the system. These classes contain error messages and help to identify the cause of the error.
* The flow package contains several sub-packages, including api, execution, mapping, and impl. The api package defines the interface for working with workflows, while the execution package contains classes for executing workflows. The impl package contains a class called stepper2flow, which takes information from the JAXB classes and puts it into objects from our classes. The mapping package contain two classes - FlowAutomaticMapping and FlowCustomMapping.
* The jaxb.schema package contains a class that receives the XML file that the user entered and inserts the information into the classes created by JAXB.
* The statistic package contains classes for storing and displaying workflow statistics. The FlowAndStepStatisticData class contains lists of type StatisticData class for flows and steps, which stores all the information needed to print the statistics data, like average running time and how many times each flow or step run.
* The steps package contains classes for defining flow steps. These classes have the necessary implementation to execute the step's logic. Additionally, this package contains the DataDefinitionDeclarationImpl class for storing flow data, the Logger class for storing logs information like the time’s log and data, and the StepDefinition class for storing steps and their implementation functions.
* The systemengine package includes the systemengine interface is the main interface that responsible to the connection between the UI module and the systemEngine module.

Packages in the JavaFX module:

* The header contains a component responsible for handling XML files. It allows the user to select an XML file and handles any errors that may occur during the file reading process.
* The flowDefinitionTab is responsible for displaying flows from a selected XML file in the HeaderController. When a flow is selected, it shows additional information about the flow, and the user can click a button to execute the flow.
* The flowExecutionTab at the top of the tab we implemented an input table into which the user can enter values. Clicking the start button executes the selected flow. During the flow execution, information about the run is displayed in the MasterDetails component. At the end of the run, the continuity table in the tab is updated. If defined in the XML file, there is a button in the continuity table that initializes the current page and enables running the selected flow with the accumulated values from the previous run as inputs.
  + In this package we have the MasterDetail package. It is responsible for displaying the content of the flow run. It provides detailed information about the flow execution, including any relevant data or logs.
* The executionsHistoryTab displays information about previous flow runs. The ExecutionHistoryEntry component implements a table that shows all the previous flow runs. Clicking on a flow selection button displays the details of the flow as they were displayed in the Flow Execution screen. At the bottom of the tab, there is a "rerun flow" button that allows the user to return to the Flow Execution screen and rerun the same flow with the same values used in the previous run.
* The StatisticsTab is responsible for receiving a DTO that contains statistics from all the runs. It displays these statistics as column charts, providing visual representations of the data.

This module also has two main Classes – desktopUI and Controller.

desktopUI and Controller are responsible for running the entire system. They who creates the scene and picks up the application. This class is responsible for reading the different fxml and uniting all the different components into one complete system.