|  |  |
| --- | --- |
| **Project title** | **image-editor** |
| **Author(s)** | **Duică Sebastian** |
| **Group** | **30424** |

# Task Description

The ongoing development of our image editing app is aimed at delivering a robust platform with an extensive set of features catered to meet diverse user needs. The app, currently in progress and utilizing Java and JavaFX, is envisioned to empower users with a versatile toolkit for manipulating and enhancing their images.

Upon accessing the app, users will encounter a preliminary interface with an open file window, offering a glimpse into the forthcoming features. The 'Open' option button will eventually allow users to load image files for editing. Once an image is selected, it will be showcased on the main canvas, awaiting a spectrum of editing possibilities.

The array of tools under development includes a hand tool, zoom tool, paint brush, eraser, custom filter, and text tools, each boasting distinct functions. For instance, the paint brush tool will enable users to draw on the image, while the eraser tool will facilitate precise removal of image elements. The upcoming custom filter tool is designed to empower users to create and apply unique filters to their images.

Furthermore, users will have the ability to apply a variety of filters, such as grayscale, sepia, invert, and custom filters, to achieve a multitude of desired effects. Post-editing, users can save their images through the 'Save' option in the evolving menu, along with options to reset the image or exit the application.

The evolving graphical user interface (GUI) is being crafted to be intuitive and user-friendly, featuring a toolbar at the top housing the tools and filters. The main canvas occupies most of the screen for seamless image editing. The 'Open,' 'Save,' 'Reset,' and 'Exit' options are expected to be found in a menu at the top left.

In summary, our image editing application, currently in development, aims to provide users with an extensive suite of functionalities. As we work towards implementing these features, a conceptual sketch is provided to offer a glimpse into the envisioned interface, allowing users to anticipate the exciting capabilities the app will soon offer.

Therefore, to visualize the future interface somehow, this sketch will aid in accurately portraying how the future project should look like:

A close-up of a grey square

Description automatically generated

Figure 1: Shows a very rough initial sketch of main interface.

# Class Discovery

The following cards explain the relationship between the classes and the functionalities of the chosen classes.

|  |  |
| --- | --- |
| **MainController** | |
| Handles the main operations in the image editor.  Provides methods to open and save files, apply filters and tools, and handle UI events.  Manages the user interface elements and their interactions. | Hand  Zoom  PaintBrush  Eraser  CustomFilter  Text  GrayscaleFilter  SepiaFilter  InvertFilter  ControllerMediator  Constants |

|  |  |
| --- | --- |
| **OpenFileController** | |
| Open a file chooser dialog to select an image file.  Store the selected file in the ControllerMediator.  Show the main stage of the application if a file is selected.  Close the current stage if a file is selected.  Set the stage for this controller. | ControllerMediator  Main  Constants |

|  |  |
| --- | --- |
| **AbstractFilter** | |
| Implement the Filter interface.  Provide a basic implementation of the apply method, which applies a color adjust effect to an image view and a canvas.  Provide a static method to deactivate the effect. | Filter |

|  |  |
| --- | --- |
| **GrayscaleFilter** | |
| Extend the AbstractFilter class.  Override the apply method to apply a grayscale effect to an image view and a canvas. | AbstractFilter |

|  |  |
| --- | --- |
| **InvertFilter** | |
| Extend the AbstractFilter class.  Override the apply method to apply a invert effect to an image view and a canvas. | AbstractFilter |

|  |  |
| --- | --- |
| **SepiaFilter** | |
| Extend the AbstractFilter class.  Override the apply method to apply a sepia effect to an image view and a canvas. | AbstractFilter |

|  |  |
| --- | --- |
| **Tool** | |
| Define the methods that all tools must implement.  Specify that a tool can be activated or deactivated.  Specify that a tool has a side menu associated with it. |  |

|  |  |
| --- | --- |
| **Filter** | |
| Define the methods that all filters must implement.  Specify that a filter can be applied to an ImageView and a Canvas. |  |

|  |  |
| --- | --- |
| **Main** | |
| Launch the JavaFX application.  Set the primary stage of the application.  Load the main scene of the application. | Constants |

|  |  |
| --- | --- |
| **OpenFile** | |
| Open a file chooser dialog to select an image file.  Store the selected file in the ControllerMediator.  Show the main stage of the application if a file is selected.  Close the current stage if a file is selected.  Set the stage for this controller. | ControllerMediator  Constants |

|  |  |
| --- | --- |
| **Hand** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the hand tool.  Implement the getSideMenu method to return the side menu associated with the hand tool. | Tool |

|  |  |
| --- | --- |
| **Zoom** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the zoom tool.  Implement the getSideMenu method to return the side menu associated with the zoom tool. | Tool |

|  |  |
| --- | --- |
| **PainBrush** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the paint brush tool.  Implement the getSideMenu method to return the side menu associated with the paint brush tool. | Tool |

|  |  |
| --- | --- |
| **Eraser** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the eraser tool.  Implement the getSideMenu method to return the side menu associated with the eraser tool. | Tool |

|  |  |
| --- | --- |
| **CustomFilter** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the custom filter tool.  Implement the getSideMenu method to return the side menu associated with the custom filter tool. | Tool |

|  |  |
| --- | --- |
| **Text** | |
| Extend the Tool interface.  Implement the activate and deactivate methods to enable and disable the text tool.  Implement the getSideMenu method to return the side menu associated with the text tool. | Tool |

# Class Diagram

The main class diagram of the project is:

A screenshot of a computer

Description automatically generated

Individual class diagrams:

A black rectangular object with white text

Description automatically generated

A black background with white text

Description automatically generated



A grey background with white text

Description automatically generated



A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

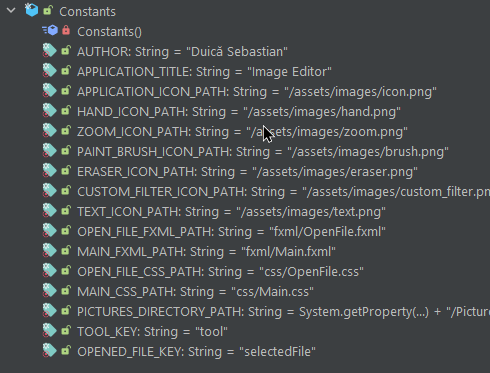
Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated



A screenshot of a computer program

Description automatically generated



A grey background with white text

Description automatically generated

A black rectangular object with white text

Description automatically generated

A black rectangular object with white text

Description automatically generated

# The class unit tests.

Testing in software development is a crucial process that involves the execution of a program or application with the intent of finding software bugs. It can also be stated as the process of validating and verifying that a software program or application or product:

1. Meets the business and technical requirements that guided its design and development.
2. Works as expected.
3. Can be implemented with the same characteristics.

In this project there are tests for the existence of assets, for the constants class and for the proper working of the controllerMediator class.

A screenshot of a computer program

Description automatically generated

A black rectangular object with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A grey background with white text

Description automatically generated