

# Module Interface Specification for OCRacle

Phillip Tran

April 17, 2025

# 1 Revision History

Date	Version	Notes
March 13, 2025	1.0	Initial Document Creation
April 7, 2025	2.0	Addressed comments from Dr. Spencer Smith and Hussein Saad, updated document to reflect implementation

## 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at <https://github.com/ptrandev/OCRacle/blob/main/docs/SRS/SRS.pdf> for symbols, abbreviations and acronyms.

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>1</b>
<b>6</b>	<b>MIS of Application Module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	3
6.3	Syntax . . . . .	3
6.3.1	Exported Constants . . . . .	3
6.3.2	Exported Access Programs . . . . .	3
6.4	Semantics . . . . .	3
6.4.1	State Variables . . . . .	3
6.4.2	Environment Variables . . . . .	3
6.4.3	Assumptions . . . . .	3
6.4.4	Access Routine Semantics . . . . .	4
6.4.5	Local Functions . . . . .	4
<b>7</b>	<b>MIS of Input Format Module</b>	<b>5</b>
7.1	Module . . . . .	5
7.2	Uses . . . . .	5
7.3	Syntax . . . . .	5
7.3.1	Exported Constants . . . . .	5
7.3.2	Exported Access Programs . . . . .	5
7.4	Semantics . . . . .	5
7.4.1	State Variables . . . . .	5
7.4.2	Environment Variables . . . . .	5
7.4.3	Assumptions . . . . .	5
7.4.4	Access Routine Semantics . . . . .	5
7.4.5	Local Functions . . . . .	6
<b>8</b>	<b>MIS of Model Output Module</b>	<b>7</b>
8.1	Module . . . . .	7
8.2	Uses . . . . .	7
8.3	Syntax . . . . .	7
8.3.1	Exported Constants . . . . .	7
8.3.2	Exported Access Programs . . . . .	7

8.4	Semantics . . . . .	7
8.4.1	State Variables . . . . .	7
8.4.2	Environment Variables . . . . .	7
8.4.3	Assumptions . . . . .	7
8.4.4	Access Routine Semantics . . . . .	7
<b>9</b>	<b>MIS of Model Training Module</b>	<b>9</b>
9.1	Module . . . . .	9
9.2	Uses . . . . .	9
9.3	Syntax . . . . .	9
9.3.1	Exported Constants . . . . .	9
9.3.2	Exported Access Programs . . . . .	9
9.4	Semantics . . . . .	9
9.4.1	State Variables . . . . .	9
9.4.2	Environment Variables . . . . .	10
9.4.3	Assumptions . . . . .	10
9.4.4	Access Routine Semantics . . . . .	10
9.4.5	Local Functions . . . . .	10
<b>10</b>	<b>MIS of Prediction Model Module</b>	<b>11</b>
10.1	Module . . . . .	11
10.2	Uses . . . . .	11
10.3	Syntax . . . . .	11
10.3.1	Exported Constants . . . . .	11
10.3.2	Exported Access Programs . . . . .	11
10.4	Semantics . . . . .	11
10.4.1	State Variables . . . . .	11
10.4.2	Environment Variables . . . . .	11
10.4.3	Assumptions . . . . .	11
10.4.4	Access Routine Semantics . . . . .	11
10.4.5	Local Functions . . . . .	12
<b>11</b>	<b>MIS of Image Preprocessing Module</b>	<b>13</b>
11.1	Module . . . . .	13
11.2	Uses . . . . .	13
11.3	Syntax . . . . .	13
11.3.1	Exported Constants . . . . .	13
11.3.2	Exported Access Programs . . . . .	13
11.4	Semantics . . . . .	13
11.4.1	State Variables . . . . .	13
11.4.2	Environment Variables . . . . .	13
11.4.3	Assumptions . . . . .	13
11.4.4	Access Routine Semantics . . . . .	13

11.4.5	Local Functions . . . . .	14
<b>12</b>	<b>MIS of Model Evaluation Module</b>	<b>15</b>
12.1	Module . . . . .	15
12.2	Uses . . . . .	15
12.3	Syntax . . . . .	15
12.3.1	Exported Constants . . . . .	15
12.3.2	Exported Access Programs . . . . .	15
12.4	Semantics . . . . .	15
12.4.1	State Variables . . . . .	15
12.4.2	Environment Variables . . . . .	15
12.4.3	Assumptions . . . . .	15
12.4.4	Access Routine Semantics . . . . .	16
12.4.5	Local Functions . . . . .	16
<b>13</b>	<b>MIS of Data Loading Module</b>	<b>17</b>
13.1	Module . . . . .	17
13.2	Uses . . . . .	17
13.3	Syntax . . . . .	17
13.3.1	Exported Constants . . . . .	17
13.3.2	Exported Access Programs . . . . .	17
13.4	Semantics . . . . .	17
13.4.1	State Variables . . . . .	17
13.4.2	Environment Variables . . . . .	17
13.4.3	Assumptions . . . . .	17
13.4.4	Access Routine Semantics . . . . .	17
13.4.5	Local Functions . . . . .	18
<b>14</b>	<b>MIS of Graphical User Interface Module</b>	<b>19</b>
14.1	Module . . . . .	19
14.2	Uses . . . . .	19
14.3	Syntax . . . . .	19
14.3.1	Exported Constants . . . . .	19
14.3.2	Exported Access Programs . . . . .	19
14.4	Semantics . . . . .	19
14.4.1	State Variables . . . . .	19
14.4.2	Environment Variables . . . . .	19
14.4.3	Assumptions . . . . .	19
14.4.4	Access Routine Semantics . . . . .	20
14.4.5	Local Functions . . . . .	20
<b>15</b>	<b>Appendix</b>	<b>22</b>

### 3 Introduction

The following document details the Module Interface Specifications for OCRacle, an optical character recognition (OCR) program for identifying Latin alphabet characters.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/ptrandev/OCRacle/>.

### 4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol  $:=$  is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

The following table summarizes the primitive data types used by OCRacle.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
bounded real	$\mathbb{R}_{\min \leq x \leq \max}$	a real number in the range $[\min, \max]$
bounded integer	$\mathbb{Z}_{\min \leq x \leq \max}$	an integer in the range $[\min, \max]$
boolean	bool	a value that is either true or false

The specification of OCRacle uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, OCRacle uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

To describe matrices, the dimensions of the matrix will be described in superscript (i.e.  $\mathbb{Z}^{w \times w}$ ). For all additional notation, reference Section 1.2 of the SRS Document. (Tran, 2025b).

### 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding Module	
	Input Format Module
	Model Output Module
	Model Training Module
Behaviour-Hiding Module	Prediction Model Module
	Application Module
	Image Preprocessing Module
	Model Evaluation Module
	Data Loading Module
Software Decision Module	Graphical User Interface Module

Table 1: Module Hierarchy



## 6 MIS of Application Module

### 6.1 Module

main

### 6.2 Uses

- Input Format Module [7](#)
- Model Output Module [8](#)
- Model Training Module [9](#)
- Graphical User Interface Module [14](#)

### 6.3 Syntax

#### 6.3.1 Exported Constants

N/A

#### 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

### 6.4 Semantics

#### 6.4.1 State Variables

N/A

#### 6.4.2 Environment Variables

- filePath: string. The path to the input image file from the user.

#### 6.4.3 Assumptions

The Graphical User Interface module [14](#) is responsible for handling all user inputs and displaying outputs. The Application module is responsible for coordinating the interaction between the GUI, Input Format, and Model Output modules.

#### **6.4.4 Access Routine Semantics**

`main()`:

- transition: The application is started and the user is able to interact with the GUI to input images and view the model's predictions.

#### **6.4.5 Local Functions**

N/A

## 7 MIS of Input Format Module

### 7.1 Module

input

### 7.2 Uses

- Image Preprocessing Module [11](#)

### 7.3 Syntax

#### 7.3.1 Exported Constants

N/A

#### 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
input	filePath: string	$\mathbb{Z}_{0 \leq x \leq 255}^{w \times w}$	DimensionsError, FormatError, FileNotFound

### 7.4 Semantics

#### 7.4.1 State Variables

N/A

#### 7.4.2 Environment Variables

- Image: The input image file to be processed by OCRacle.

#### 7.4.3 Assumptions

N/A

#### 7.4.4 Access Routine Semantics

input(filePath):

- output: imageMatrix := preprocessing(filePath). After ensuring that all exceptions are handled, call the preprocessing module [11](#) to perform the necessary transformations on the input image. <sup>1</sup>

---

<sup>1</sup>By splitting up the file handling concerns with the preprocessing concerns, we are able to ensure that AC3 and AC5 from the MG ([Tran, 2025a](#)) are met. This separation of concerns allows for better modularity and maintainability of the code. For instance, if we wanted to be able to support multiple preprocessing methods or allow input of files other than images.

- exceptions:
  - A `DimensionsError` exception is thrown if the input image exceeds the bounds of the  $n_{min}$ ,  $n_{max}$ ,  $m_{min}$ , and  $m_{max}$  parameters as specified by Table 4 in the SRS ([Tran, 2025b](#), 4.2.6).
  - A `FormatError` exception is thrown if the input image is not one of the supported formats specified by R1 in the SRS ([Tran, 2025b](#), 5.1.1).
  - A `FileNotFound` exception is thrown if the input file does not exist.

#### 7.4.5 Local Functions

N/A

## 8 MIS of Model Output Module

### 8.1 Module

output

### 8.2 Uses

- Prediction Model Module [10](#)

### 8.3 Syntax

#### 8.3.1 Exported Constants

- LABELS: ['A'...'Z']

#### 8.3.2 Exported Access Programs

Name	In	Out	Exceptions
output	imageMatrix: $\mathbb{Z}_{0 \leq x < 255}^{w \times w}$	char, $\mathbb{R}_{0 \leq x \leq 1}^{1 \times L}$	-

### 8.4 Semantics

#### 8.4.1 State Variables

N/A

#### 8.4.2 Environment Variables

N/A

#### 8.4.3 Assumptions

We assume that the imageMatrix input has been processed by the Input Format Module [7](#) before being passed to the Model Output Module.

#### 8.4.4 Access Routine Semantics

output(imageMatrix):

- output:
  - probabilityDistribution := predict(imageMatrix). Call the prediction model module [10](#) to generate a probability distribution for the input image.

- `prediction := LABELS[argmax(probabilityDistribution)]`. Return the character representing the highest confidence value in the probability distribution. When there is a tie, the first most confident character is returned.

## 9 MIS of Model Training Module

### 9.1 Module

train

### 9.2 Uses

- Prediction Model Module [10](#)
- Data Loading Module [13](#)

### 9.3 Syntax

#### 9.3.1 Exported Constants

N/A

#### 9.3.2 Exported Access Programs

Name	In	Out	Exceptions
train	-	$\mathbb{R}^{m \times n}$	-

### 9.4 Semantics

#### 9.4.1 State Variables

- dataset:  $(\mathbb{Z}_{0 \leq x \leq 255}^{w \times w}, \text{char})[]$ . The dataset of images and their corresponding labels used for training the model.
- epochs:  $\mathbb{N}$ . The number of epochs to train the model.
- batchSize:  $\mathbb{N}$ . The number of images to train on in each batch.
- timeStep:  $\mathbb{R}$ . The time step for the ADAM optimizer.
- $\alpha$ :  $\mathbb{R}$ . The rate at which the model learns.
- $\epsilon$ :  $\mathbb{R}$ . A small value to prevent division by zero.
- $m_t$ :  $\mathbb{R}$ . The first moment for the ADAM optimizer.
- $v_t$ :  $\mathbb{R}$ . The second moment for the ADAM optimizer.
- $\beta_1$ :  $\mathbb{R}$ . The  $\beta_1$  parameter for the ADAM optimizer.
- $\beta_2$ :  $\mathbb{R}$ . The  $\beta_2$  parameter for the ADAM optimizer.

- $g_t$ :  $\mathbb{R}^{m \times n}$ . The gradient of the model.
- $\theta_t$ :  $\mathbb{R}^{m \times n}$ . The weights of the model.

#### 9.4.2 Environment Variables

N/A

#### 9.4.3 Assumptions

N/A

#### 9.4.4 Access Routine Semantics

`train()`:

- transition: The model is trained on the dataset using the hyperparameters  $\alpha$ ,  $\beta_1$ ,  $\beta_2$ ,  $\epsilon$ ,  $m_t$ ,  $v_t$ ,  $g_t$ ,  $\theta_t$ , epochs, batchSize, and timeStep. The ADAM optimizer is used to minimize the cross-entropy loss as specified by TM:ADAM ([Tran, 2025b](#), 4.2.2) and IM1 in the SRS ([Tran, 2025b](#), 4.2.4). For each step, theta is updated using the gradient of the model.
- output:  $\theta$ . Once the model has been trained and loss has been minimized, the weights of the model are returned.

#### 9.4.5 Local Functions

N/A



## 10 MIS of Prediction Model Module

### 10.1 Module

model

### 10.2 Uses

N/A

### 10.3 Syntax

#### 10.3.1 Exported Constants

N/A

#### 10.3.2 Exported Access Programs

Name	In	Out	Exceptions
predict	imageMatrix: $\mathbb{Z}_{0 \leq x < 255}^{w \times w}$	$\mathbb{R}_{0 \leq x \leq 1}^{1 \times L}$	-

### 10.4 Semantics

#### 10.4.1 State Variables

- model:  $\mathbb{R}^{m \times n}$ . The representation of the model's weights for the prediction model.

#### 10.4.2 Environment Variables

N/A

#### 10.4.3 Assumptions

We assume that the input image has been preprocessed as specified in the SRS before being passed to the prediction model.

#### 10.4.4 Access Routine Semantics

predict(imageMatrix):

- output: probabilityDistribution. Where each element in the probability distribution represents the model's confidence in the corresponding character. The model is used to generate a probability distribution for the input image using a CNN as specified by GD3 in the SRS (Tran, 2025b, 4.2.3). GD3 provides a high-level description of the model's architecture, but the implemented model may differ.

### 10.4.5 Local Functions

N/A

## 11 MIS of Image Preprocessing Module

### 11.1 Module

preprocessing

### 11.2 Uses

N/A

### 11.3 Syntax

#### 11.3.1 Exported Constants

N/A

#### 11.3.2 Exported Access Programs

Name	In	Out	Exceptions
preprocessing	file : string	$\mathbb{Z}_{0 \leq x < 255}^{w \times w}$	-

### 11.4 Semantics

#### 11.4.1 State Variables

N/A

#### 11.4.2 Environment Variables

N/A

#### 11.4.3 Assumptions

All input data has been validated by the Input Format Module before being processed by the Image Preprocessing Module.

#### 11.4.4 Access Routine Semantics

preprocessing(file):

- output: imageMatrix. The input file has been transformed such that it conforms to the preprocessing requirements as specified by A3 in the SRS ([Tran, 2025b](#), 4.2.1). First bicubic interpolation is used to resize the image to the required dimensions as specified by TM:BI in the SRS document ([Tran, 2025b](#), 4.2.2). Then the image is converted to grayscale and the pixel values are normalized to be between 0 and 1 inclusive as specified by TM:N in the SRS document ([Tran, 2025b](#), 4.2.2).

#### 11.4.5 Local Functions

N/A

## 12 MIS of Model Evaluation Module

### 12.1 Module

evaluation

### 12.2 Uses

- Model Output Module [8](#)
- Prediction Model Module [10](#)
- Data Loading Module [13](#)

### 12.3 Syntax

#### 12.3.1 Exported Constants

N/A

#### 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
evaluate	-		-

### 12.4 Semantics

#### 12.4.1 State Variables

N/A

#### 12.4.2 Environment Variables

N/A

#### 12.4.3 Assumptions

We assume that the Prediction Model Module [10](#) has been trained before being used to evaluate the model's performance. The evaluation is performed on the test dataset extracted from the Data Loading Module [13](#).

#### 12.4.4 Access Routine Semantics

evaluate():

- output:

- $\text{loss} := -\sum_{i=1}^n \text{target}_i \log(\text{prediction}_i)$ . The cross entropy loss is calculated for the model's predictions and the target values as specified by TM:CELF in the SRS ([Tran, 2025b](#), 4.2.2).
- $\text{accuracy} := \frac{1}{n} \sum_{i=1}^n \text{target}_i \times \text{prediction}_i$ . The accuracy is calculated for the model's predictions as described in the VnV Plan ([Tran, 2025c](#)). This fulfills NFR1 in the SRS ([Tran, 2025b](#)).
- $\text{confusionMatrix} := \text{confusionMatrix}_{i,j} = \sum_{k=1}^n \text{target}_{k,i} \times \text{prediction}_{k,j}$ . A confusion matrix is generated for the model's predictions and the target values.
- $\text{classAccuracy} := \frac{1}{n} \sum_{i=1}^n \text{target}_i \times \text{prediction}_i$ . The class accuracy is calculated for each class in the confusion matrix and sorted by accuracy descending.

#### 12.4.5 Local Functions

N/A

## 13 MIS of Data Loading Module

### 13.1 Module

data

### 13.2 Uses

N/A

### 13.3 Syntax

#### 13.3.1 Exported Constants

N/A

#### 13.3.2 Exported Access Programs

Name	In	Out	Exceptions
extractTestSamples	-	$\mathbb{R}_{0 \leq x \leq 1}^{w \times w}[], \mathbb{Z}[]$	-
extractTrainingSamples	-	$\mathbb{R}_{0 \leq x \leq 1}^{w \times w}[], \mathbb{Z}[]$	-

### 13.4 Semantics

#### 13.4.1 State Variables

N/A

#### 13.4.2 Environment Variables

N/A

#### 13.4.3 Assumptions

N/A

#### 13.4.4 Access Routine Semantics

extractTestSamples():

- output: (images, labels) := processSamples(testImages, testLabels). A tuple containing the test images and their corresponding labels. The test samples are extracted from the EMNIST letters dataset.

extractTrainingSamples():

- output:  $(\text{images}, \text{labels}) := \text{processSamples}(\text{trainingImages}, \text{trainingLabels})$ . A tuple containing the training images and their corresponding labels. The training samples are extracted from the EMNIST letters dataset.

#### 13.4.5 Local Functions

`processSamples()`:

- output:  $(\text{images}, \text{labels})$ . The images are normalized to have a pixel range between 0 and 1 inclusive, as specified by the SRS document ([Tran, 2025b](#)). The labels are scaled to start at 0 instead of 1 to align with the model architecture's output as specified by the SRS document ([Tran, 2025b](#)).



## 14 MIS of Graphical User Interface Module

### 14.1 Module

gui

### 14.2 Uses

- Hardware-Hiding Module

### 14.3 Syntax

#### 14.3.1 Exported Constants

N/A

#### 14.3.2 Exported Access Programs

View the Appendix [15](#) for more information the full range of interactions that the Graphical User Interface Module can have with the user.

Name	In	Out	Exceptions
displayOutput	-	-	-
acceptInput	-	-	-

### 14.4 Semantics

#### 14.4.1 State Variables

N/A

#### 14.4.2 Environment Variables

- displayWindow: (dimensions:  $\mathbb{Z}_+ \times \mathbb{Z}_+$ )
- keyboard: (keypress: char)
- mouse: (location:  $\mathbb{Z}_+ \times \mathbb{Z}_+$ , click: bool)

#### 14.4.3 Assumptions

This behavior will be significantly abstracted by the use of Python Notebook, which will handle all UI elements and interactions with the user. As a result, the GUI modules does not need to have any pre-defined state variables or access routines.

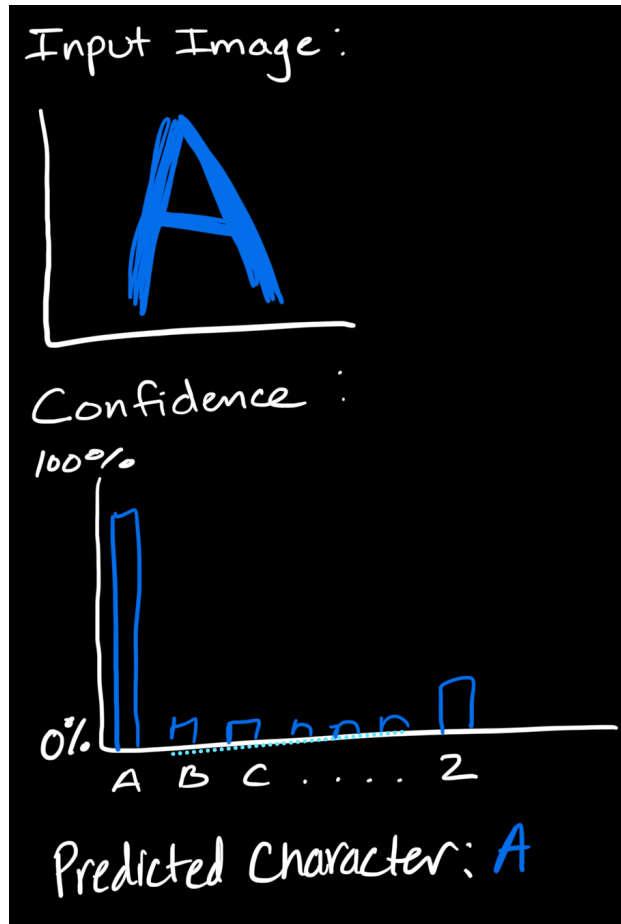


Figure 1: Example of the `displayOutput()` function. It will display the model's prediction and the probability distribution to the user in a graphical format.

#### 14.4.4 Access Routine Semantics

`displayOutput()`:

- transition: The output of the model is displayed to the user in the GUI.

`acceptInput()`:

- transition: The user is able to input images to the model in the GUI.

#### 14.4.5 Local Functions

N/A

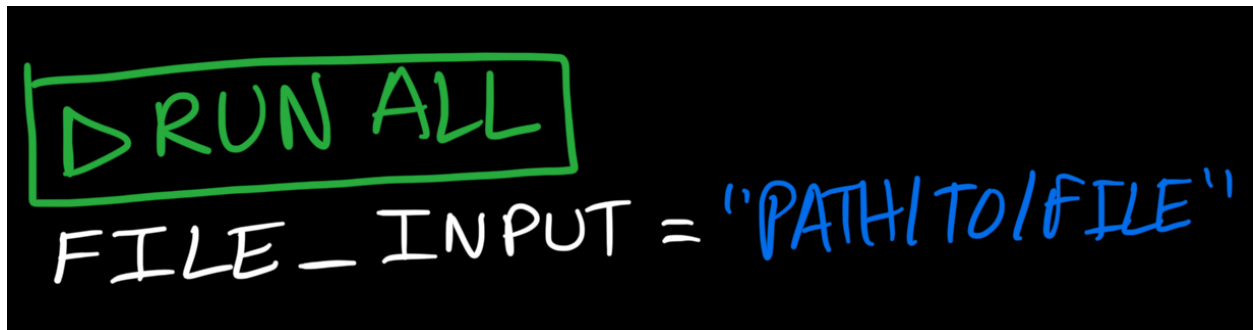


Figure 2: Example of the `acceptInput()` function. It allows users to input a file path and run the model on the input image.

## References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.
- Phillip Tran. Module guide. <https://github.com/ptrandev/OCRacle/blob/main/docs/Design/SoftArchitecture/MG.pdf>, 2025a.
- Phillip Tran. System requirements specification. <https://github.com/ptrandev/OCRacle/blob/main/docs/SRS/SRS.pdf>, 2025b.
- Phillip Tran. Verification and validation plan. <https://github.com/ptrandev/OCRacle/blob/main/docs/VnVPlan/VnVPlan.pdf>, 2025c.

## 15 Appendix

- The Python Notebook documentation can be found at <https://jupyter-notebook.readthedocs.io/en/latest/user-documentation.html>. This informs the full range of interactions that the Graphical User Interface Module 14 can have with the user.
- Generative AI Acknowledgement: ChatGPT-4o was used to generate valid LaTeX for mathematical expressions in this document.