**Design doc - Adi Rosenthal**

**Introduction**

***Scope***

The goal of this project is implementation of a software that would be able to convert a picture (handwritten or printed text) in Hebrew to machine encoded text.

This prosses widely used as a form of data entry from printed paper data records and digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes.

For example, students who prefer handwritten documents can transform these into digital documents with endless applications (sharing, search and update more easily and conveniently).

***Overview***

The designdocument will contain the following information:

1. General Flow of the software.
2. The components architecture design and description of the structures - Attributes and relationships between data objects.
3. Description about the UI in the program.
4. Explanation about the tests that will apply over the software – including alerts and logs.

***Terminology***

* **OCR** - Optical character recognition is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text.
* **Tesseract** - Tesseract is an optical character recognition. It is free software, released as open source and development has been sponsored by Google since 2006.
* **OpenCV** - is a library of programming functions mainly aimed at real-time computer vision.

**Software design description**

***General Flow***

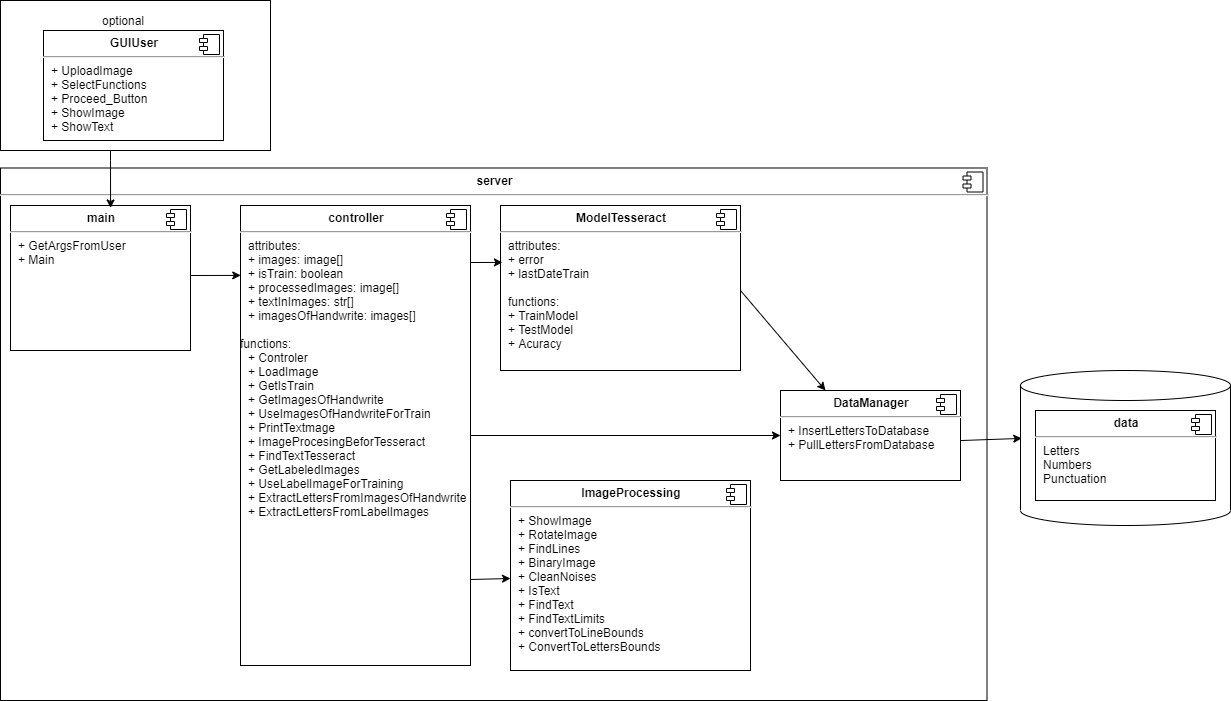
The program’s users will be able to insert an image (using the program’s GUI), click on the “run” button and extract all the text that in the image, particular in Hebrew.

The user will be able to watch the manipulated image that used for the ML prosses and will be able to save the text of the result.

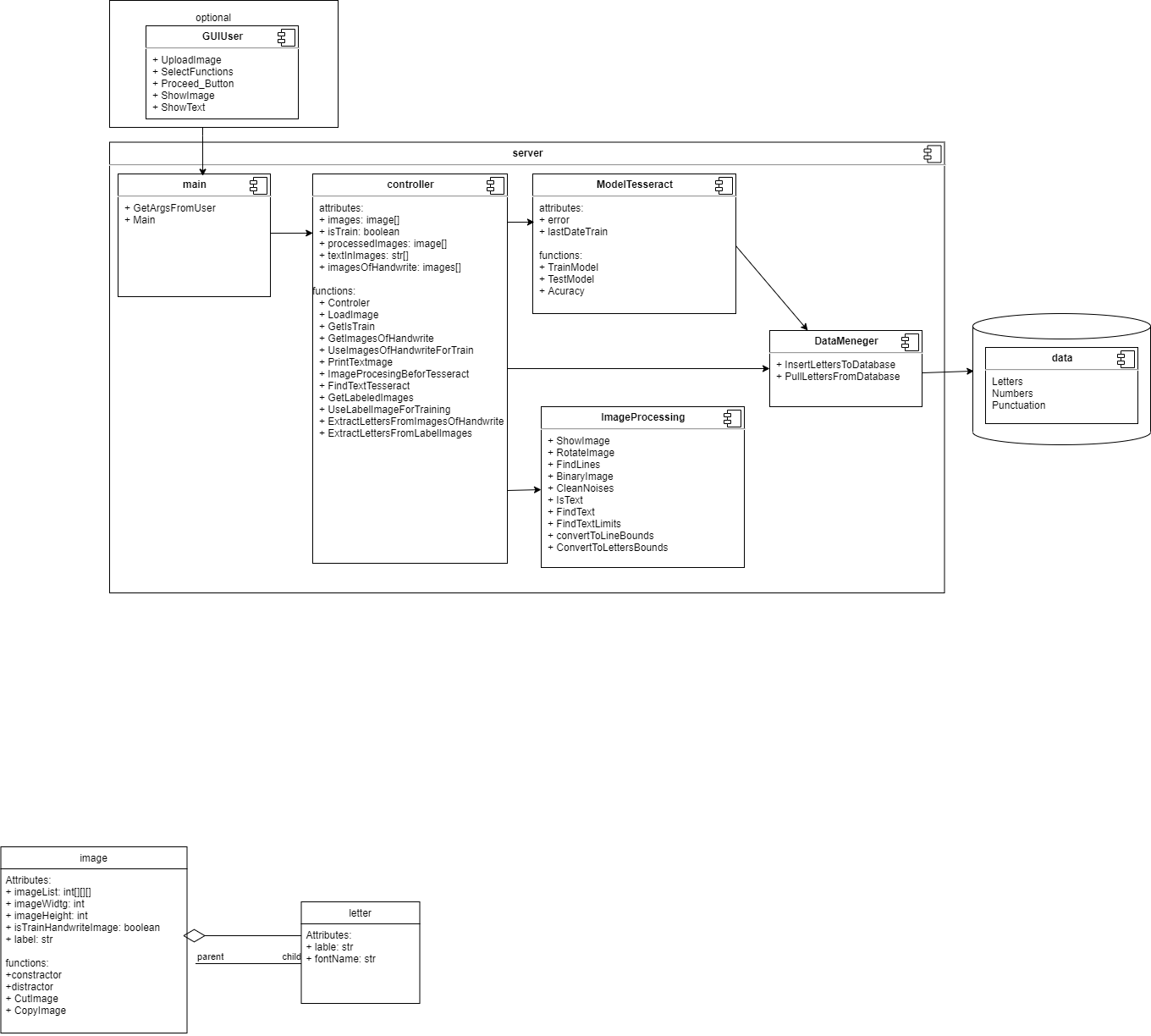
The program may send alert if something got wrong.

The text recognition prosses will include image processing including: remove noises from the image, rotate the image, classify the pixel values. After the image processing the program will use “Tesseract” neuron network (after training it) and extract the text from it.

***Describe here all the relevant components***

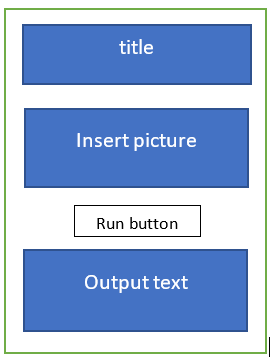


System entities:



***UI - optional***

Bonus option - using one of the python GUI classes like Tkinter , pyqt, Tk or Python Xlib to create user interface in python that will look like this diagram :

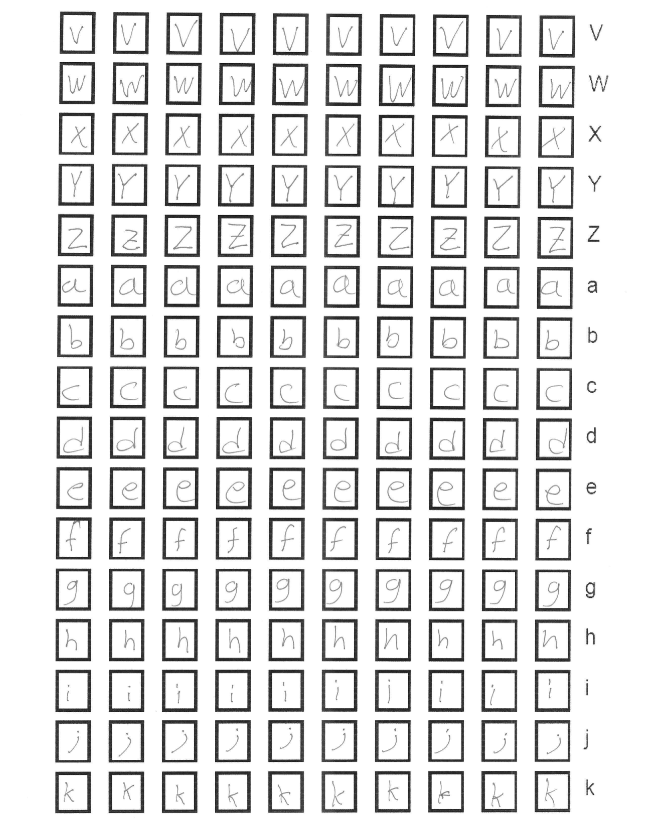


***Data***

We use data (handwritten letters or printed letters) in Hebrew, as an input for training process of “Tesseract” model.

In order to collect this data, we can use 2 different processes:

1. The user inserts images with text and insert also labels (txt files) for those images.
2. The user prints and scanned pages where he fills his handwrite for each letter, like:

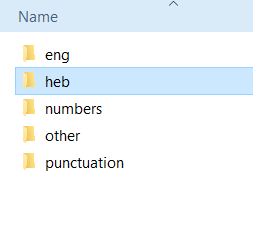


Then these letters well extract from the image and will insert to the data folders using for training.

In addition, we are also need images with text and their labels (txt files) so we could calculate the accuracy of the model after the training process.

After all the training the user will insert image of text without labels and the software will return her prediction for the text.

In the Database we save all the letters image with their labels, example:



***Integration with other systems***

The software integrates with the “Tesseract” software that needs to be installed on the server. This service is based on neural network that need to be trained on the Database.

OpenCV is a library in python that will use for image processing in the project.

**Test and Monitor**

***Tests***

Basic network testing:

* + False – negative
  + False – positive
  + True – negative
  + True – positive
  + Accuracy / Precision

The software should be testing by the following parameters:

1. “Basic network testing” on different kinds of fonts (printed text).
2. “Basic network testing” on different kinds of handwritings.
3. “Basic network testing” on different processes of image manipulations (\processing).
4. Integration test - check that all the components works well with each other.
5. Unit test for all the code’s functions.

***Logs files***

There will be 4 kinds of logs files:

1. Network log – contain all the prosses that been done on the network: training and testing processes (date, input, output -successes \ fail, username), network testing result for the network.
2. Data log – contain all the “insert data process”: size, the resource, date, username.
3. Event log – document all the events during the software running: save function details - name, calling time, input and the output. This kind of log file save only the results from the last day\week.
4. Running software log - document every time the software is running: save the inputs and outputs - successes \ fail, date and username.

***Alerts***

The software should popup an alert if the following happened:

1. The user inserts a wrong input (according to the “input image rules”)
2. The network failed to recognized text (alert: “Failed” + the reason why it is failed)
3. Runtime error - the code failed.