RAMAIAH
Institute of Technology

SRS DOCUMENT FOR OCR SOFTWARE REVIEW-2

SUBMITTED BY

NAME	USN	SIGNATURE
S D HRUDHAY	1MS20CS098	
SAFWAN G A	1MS20CS099	
SANSKAR R GONDKAR	1MS20CS107	

As part of the Course **Software Engineering - CS46**

SUPERVISED BY

Faculty

Dr. Mohan Kumara S

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

Bangalore – 560054

RAMAIAH
Institute of Technology

TABLE OF CONTENTS

1. SOFTWARE ARCHITECTURE

The design of the application is highly important in order to fulfil the requirements and functionalities of the project.

Architecture of the System:

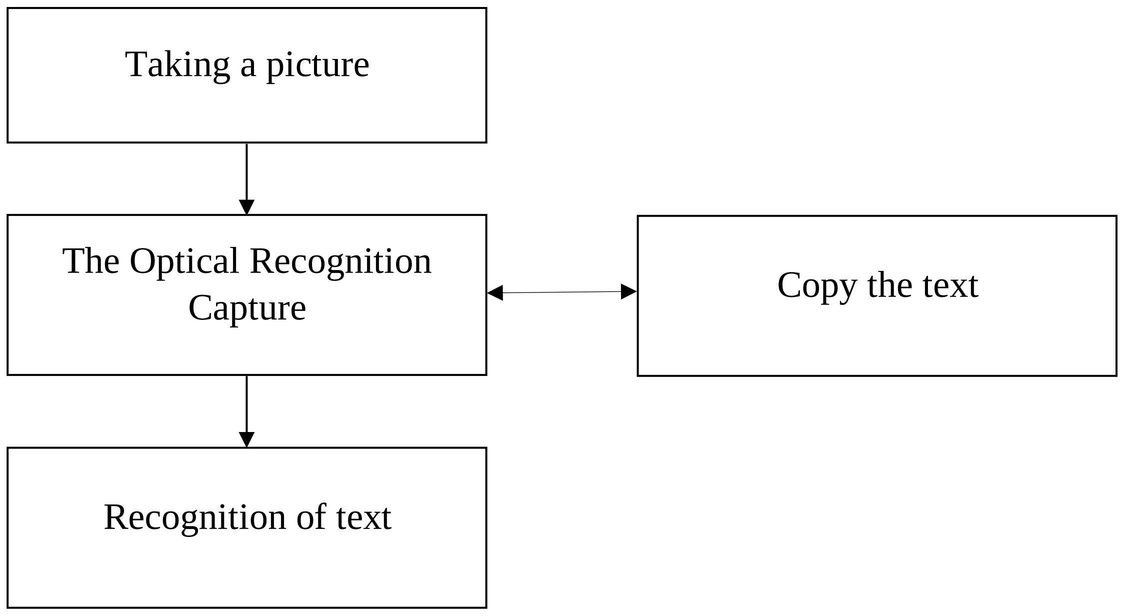

Figure 1: Structure design of OCR

The system is modelled as follows:

The interface of the system is separated from its logic by using a Model View Controller. The Model contains the data while the interface constructs the view of the architecture. The controller matches between the model and the view. All communications between the model and the view will go through the controller. Moreover, the interface doesn't follow the logic of the application which makes the architecture of the system easier.

Also, for the user class which has many functions such as presscaptureImageBotton() and pressImportImageBotton() handles the interface's process without dealing with the application's logic.

Then, the system's controller contains handlers for the system's functionality which is composed into modules. In other words, the handler of CaptureImage is used in Capture image module, the handler of ImportImage is used inImport Picture module, and the RecognizeText is used in the UserApplication module. This controller calls the functions, but do not process them.

Finally, the application's model processes the functions of all invoked functions. This model is made of CaptureImage, ImportImage, RecognizeText, and UserApplication classes that summarize the logic of the application. These classes have different functionalities and process requests of the aforementioned view and controller, retrieve the result, and give it back to the controller.

Android Mobile Application Description:

Android mobile OCR is a software development kit (SDK) based on OCR technologies. Its powerful image processing algorithms enable mobile devices to perform highly accurate text recognition.

Optimized for efficiency and featuring comprehensive language support, the application is ideal for wide range of individuals and users of android mobile phones who seek to convert their scanned paper documents, PDF files or images captured by their android mobile phones into editable and searchable data.

Using the code OCR technology, highly accurate text recognition functionality can be included in applications for tablets, Smart phones and other mobile devices using android. Mobile devices using iOS is out of our scope and can be done as a future work.

For the Android, it needs to use the procedures of the Android operating system. In other words, in order to have some tasks done, the system's implementation uses some libraries, so the application will interact with these libraries as well.

Architecture Diagram:

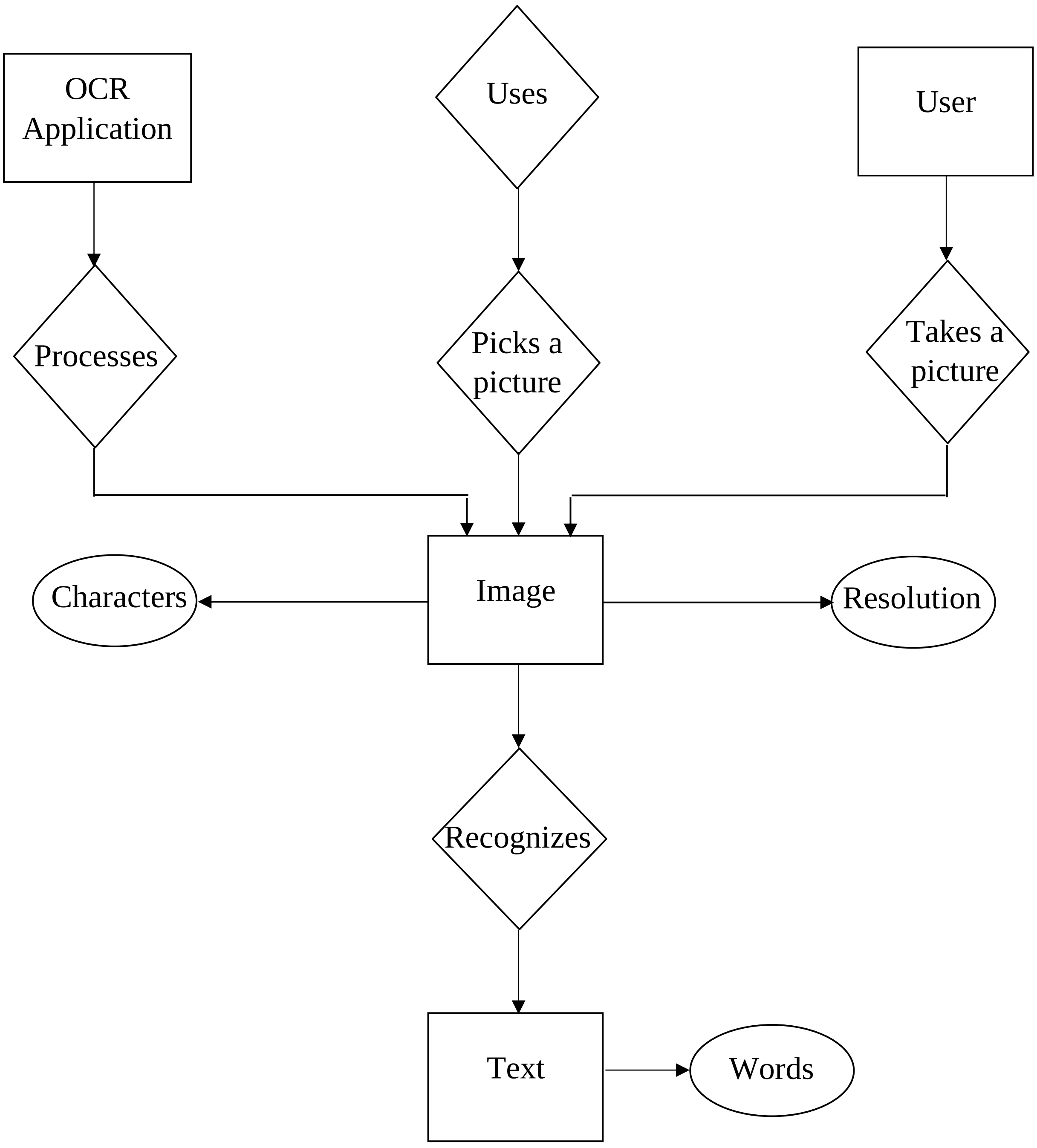

Figure 2: Architecture of OCR system

Description of the Attributes:

1. **User:** The one that will interact with the Optical Capture Recognition (OCR).

2. **The Image:** Taken: must have a certain resolution and must contain text. Selected from the phone: must contain text.

3. **The Text:** must have words.

4. **Words:** English Alphabet.

5. Optical Capture Recognition Application.

Description of the relationships between the modules:

1. **User-OCR Application:** It is one to one relationship because only one user interacts with the application at a time.

2. **OCR Application-Image:** The application is able to process one image at a time.

3. **User-Image:** The user can either take a picture or choose one from the phone directory.

4. **Image-Text:** The image must contain text.

5. **Text-Words:** The text can contain many words.

6. **Words-Characters:** Each word can have any number of English Alphabet's characters.

7. **User-Text:** The user can copy, paste, or select the whole text or just a part of it.

The software components and technology used in this project are:

Technology enablers:

- Java as an object-oriented programming language.
- Android SDK(Software development Kit)
- Android Studio.

Operating Systems:

- Windows
- Android OS

Hardware Components:

- Personal Computers
- Android based mobile phone with high resolution.

Product Description:

OCR also called Optical Character Reader is a system that provides a full alphanumeric recognition of printed or handwritten characters at electronic speed by simply scanning the form. Forms containing characters images can be scanned through scanner and then recognition engine of the OCR system interpret the images and turn images of handwritten or printed characters into ASCII data (machine-readable characters).

Therefore, OCR allows users to quickly automate data capture from forms, eliminate keystrokes to reduce data entry costs and still maintain the high level of accuracy required in forms processing applications.

The technology provides a complete form processing and documents capture solution. Usually, OCR uses a modular architecture that is open, scalable and workflow controlled. It includes forms definition, scanning, image pre-processing, and recognition capabilities.

This project has two motivations, a desktop application and Android based mobile application. Scanned documents, pictures stored in Android devices, and pictures taken by an Android device containing a text are scanned by this OCR.

This application will allow its users to retrieve text from these documents to make it editable or to reuse it to perform other actions. For example, people can scan any ID Card, and store its data in their phones directly, or scan books or any important article found in a newspaper and retrieve the needed information and store them in their devices in few minutes.

The Communication Description:

This application doesn't require any network connection. It will work as a device in-house application. This is very useful since sometimes people need to extract data from documents while internet is not working, so this application will allow them to so.

Constraints of the Implementation and Design:

The application has been developed for Android based devices with high camera's resolution. Also, the printed text without skew is the mainly focus of the application because of time constraint, but it can be added later on.

2. TESTING

Introduction:

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest.

Testing is the process of exercising software with the intent of finding (and ultimately correcting) errors. In fact, because Web-based systems and applications reside on a network and interoperate with many different operating systems, browsers (residing on a variety of devices), hardware platforms, communications protocols, and "backroom" applications, the search for errors represents a significant challenge

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Testing is a set of activities that can be planned in advance and conducted systematically.

Software testing is one element of a broader topic that is often referred to as verification and validation (V&V). Verification refers to the set of tasks that ensure that software correctly implements a specific function. Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Boehm states this way:

Verification: "Are we building the product right?"

Validation: "Are we building the right product?"

System Testing:

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

Interface Testing:

Interface Testing is defined as a software testing type which verifies whether the communication between two different software systems is done correctly. A connection that integrates two components is called interface. This interface in a computer world could be anything like API's, web services, etc. Testing of these connecting services or interface is referred to as Interface Testing.

Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. Test case design techniques that focus on inputs and outputs are more prevalent during integration, although techniques that exercise specific program paths may be used to ensure coverage of major control paths.

Validation Testing:

Validation testing provides final assurance that software meets all informational, functional, behavioural, and performance requirements. The last high-order testing step falls outside the boundary of software engineering and into the broader context of computer system. engineering. Software, once validated, must be combined with other system elements (e.g., hardware, people, databases).

Unit Testing:

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components.

Smoke Testing:

Smoke Testing is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program are working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application so that the QA team does not waste time installing and testing the software application. In Smoke Testing, the test cases chose to cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.

Regression Testing:

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the new code changes are done.

Acceptance Testing:

Acceptance is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment. The main purpose of this testing is to validate the end to end business flow. It does NOT focus on cosmetic errors, Spelling mistakes or System testing. This testing is carried out in a separate testing environment with production like data setup. It is a kind of black box testing where two or more end users will be involved.

Sanity Testing:

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing. The objective is "not" to verify thoroughly the new functionality but to determine that the developer has applied some rationality (sanity)while producing the software.

Black Box Testing:

A software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Test design techniques include Equivalence partitioning, Boundary Value Analysis, Cause-Effect Graphing. Black Box Testing, also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

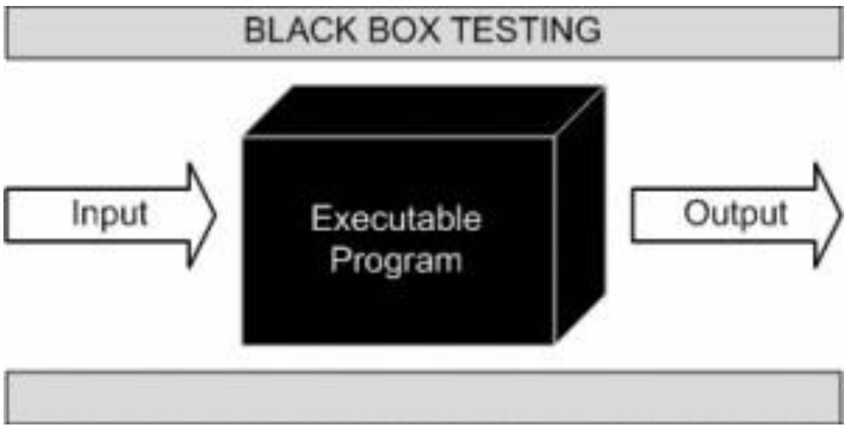

Figure 3: Black Box Testing

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions.
- Interface errors.
- Errors in data structures or external database access.
- Behaviour or performance errors.
- Initialization and termination errors.

Black box test design technique:

Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure. Black Box Testing method is applicable to the following levels of software testing:

- System Testing.
- Integration Testing.
- Acceptance Testing.

The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use.

White Box Testing:

White box testing techniques analyse the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Test design techniques include Control flow testing, Data flow testing, Branch testing, Path testing.

- White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system. White-box testing is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases. Using white-box testing methods, you can derive test cases that: Guarantee that all independent paths within a module have been exercised at least once.

- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to ensure their validity.

Basis path testing is a white-box testing technique first proposed by Tom McCabe. The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

Working process of white box testing:

Input: Requirements, Functional specifications, design documents, source code.

Processing: Performing risk analysis for guiding through the entire process.

Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.

Output: Preparing final report of the entire testing process.

White-box testing's basic procedures require the tester to have an in-depth knowledge of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analysed for test cases to be created.

TESTCASE:

Description	**Image State**	**Expected Result**
Captures the image	It has some text and clearly visible	Capture of image successful
Captures the image	If it has some text and the image is not clearly visible	Go back to capture the image
Capture the image	If the image is blank	Go back to capture the image
After the capture of image	If the image has any unnecessary thing like shapes, dots etc.	It won't recognize that one
After the capture of image	If it won't recognize the text	Try to search with the text which is more similar
After the capture of image	If the text is partially recognized	Try to search with the text which is very similar to remaining part
After the capture of image	If the text is completely recognized	Recognition of text is successful

3. CONCLUSION

Traditional optical character recognition has been available for over a century now, and are in the process of being suspended by neural network alternatives, which have greater accuracy, as well as the ability to recognise handwriting. Nevertheless, OCR was necessity in the process. Some techniques were far better than the others on different scripts. Datasets available globally fail to incorporate real-life scenarios occurring in things like uneven spacing, distorted strokes etc, giving an edge to OCR over their neural network counterparts at the start. OCR being extensive, was originally capable to be created by only large companies, but machine learning has made it accessible to all to create their own. In any case, OCR is now a mature technique that can be reliably employed in many scenarios.

4. REFERENCES

- Roger S Pressman and Bruce R. Maxim, Software Engineering: A Practitioner's Approach, 8/e, New York: McGraw-Hill, 2015.
- F. Solimanpour, J. Sadri, and C. Y. Suen, ‘‘Standard databases for recognition of handwritten digits, numerical strings, legal amounts, letters and dates in farsi language,’’ in Proc. 10th Int. Workshop Frontiers Handwriting Recognit., 2006, pp. 1–6.
- Software Requirement Specification (SRS) Format - GeeksforGeeks
- S. Deshmukh and L. Ragha, ‘‘Analysis of directional features–stroke and contour for handwritten character Recognition,’’ in Proc. IEEE Int.Advance Comput. Conf. (IACC), Mar. 2009, pp. 1114–1118.
- https://aws.amazon.com/what-is/ocr/
- K. G. Joe, M. Savit, and K. Chandrasekaran, ‘‘Offline character recognition on segmented handwritten kannada characters,’’ in Proc. Global Conf. Advancement Technol. (GCAT), Oct. 2019,pp. 1–5.
- R. Hussain, A. Raza, I. Siddiqi, K. Khurshid, and C. Djeddi, ‘‘A comprehensive survey of handwritten document benchmarks: Structure, usage and evaluation,’’ EURASIP J. Image Video Process., vol. 2015, no. 1,p. 46, Dec. 2015.
- M. Zimmermann and H. Bunke, ‘‘Automatic segmentation of the IAM off-line database for handwritten English text,’’ in Proc. 16th Int. Conf.Pattern Recognit., vol. 4, 2002.
- https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition
- U. R. Babu, A. K. Chintha, and Y. Venkateswarlu, ‘‘Handwritten digit recognition using structural, statistical features and k-nearest neighbor classifier,’’ Int. J. Inf. Eng. Electron. Bus., vol. 6, no. 1, p. 62,2014.
- S. Ahlawat and R. Rishi, ‘‘Off-line handwritten numeral recognition using hybrid feature set—A comparative analysis,’’ Procedia Comput. Sci.,vol. 122, pp. 1092–1099, Jan. 2017.
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9151144