HAW HAMBURG

BACHELORTHESIS
Shukhrat Khaydarov

Detection, Identification and Automated Solution of Logic Puzzles on a Mobile Application using Image Processing, Classification and Backtracking algorithms

FACULTY OF COMPUTER SCIENCE AND ENGINEERING
Department of Information and Electrical Engineering

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

HAMBURG UNIVERSITY
OF APPLIED SCIENCES

Hochschule für Angewandte
Wissenschaften Hamburg

Shukhrat Khaydarov

Detection, Identification and Automated Solution of Logic Puzzles on a Mobile Application using Image Processing, Classification and Backtracking algorithms

Bachelor Thesis based on the examination and study regulations
for the Bachelor of Engineering degree programme
Bachelor of Science Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg
Supervising examiner: Prof. Dr.-Ing. Marc Hensel
Second examiner: Prof. Dr. Annabella Rauscher-Scheibe

Day of delivery: 30. July 2020

Shukhrat Khaydarov

Title of Thesis

Detection, Identification and Automated Solution of Logic Puzzles on a Mobile Application using Image Processing, Classification and Backtracking algorithms

Keywords

Android Development, Logic Puzzles, Image Processing, Classification, Backtracking

Abstract
Inside this report the development of Android Application for Logic Puzzles Solver with Image Processing is described ...

Shukhrat Khaydarov

Thema der Arbeit

Erkennung, Identifizierung und automatisierte Lösung von Logikrätseln in einer mobilen Anwendung mithilfe von Bildverarbeitungs-, Klassifizierungs- und Backtracking-Algorithmen

Stichworte

Android-Anwendungsentwicklung, Logik-Rätsel, Bildverarbeitung, Klassifizierung, Backtracking

Kurzzusammenfassung

In diesem Arbeit wird die Entwicklung der Android-Anwendung fuer Logik-Rätsel Solver mit Bildverarbeitung beschrieben ...

Contents

List of Figures

List of Tables

1 Introduction

Logic puzzles are of the great assistance for the people to sharpen their mental ability skills such as numerical memory, attention to details and thinking process. It is well-known that Sudoku brings state of calmness and sense of order due to the fact that this game gives a chance to change a focus and relax from the surrounding world. It is widely practiced to include Sudoku puzzles in the daily routine owing to the reason that solving Sudoku refreshes mind and helps to complete other tasks with restored state of mind and energy.

A study conducted by the University of Exeter and King's College London [1] was attended by 19,000 people. Scientists previously presented their findings on the benefits of verbal puzzles at a conference in 2018. In the previously mentioned study, it is noted that the new PROTECT study builds on these discoveries, complemented by yet another: numerical puzzles have the same effect on the brain.

Scientists interviewed participants online how often they solve crossword and Sudoku puzzles, and then suggested they take a series of tests for cognitive functions to test brain function. It turned out that those people who regularly solve puzzles showed the best results in tasks on attention, thinking and memory.

Dr. Ann Corbett of Exeter University, who conducted the current study, notes:

"We found that the more regularly people solve puzzles such as crosswords and Sudoku, the better they get tests for memory, attention and thinking. Particularly noticeable improvements in the accuracy and speed of their decisions. In some cases, the changes were very remarkable: in solving problems, people showed results that people usually have on average eight years younger. We cannot say that solving puzzles uniquely reduces the risk of developing dementia at a later age, but this study confirms that regular solving of verbal and numerical puzzles helps the brain to work longer and better."

[1] https://www.exeter.ac.uk/news/featurednews/title_716265_en.html (accessed: 12.07.2020)

Logic puzzles can be of different types and the scope of this thesis will concentrate only on Sudoku Game. The scope can be expanded as much as it is preferred, e.g. this app can give solution to the chess game.

1.1 Motivation

The motivation for this Project is to discuss the insights about means for image processing on Android mobile device platform and make the product prototype, which shall focus on the following features:

- Portable: The product shall be attractive to user in a sense that user can use it in any place with no restriction on weight, size or any other very specific requirement
- Affordable: Costs related to the development and usage of application shall be zero, thus the user should only have a single mobile hardware device, where the software will run.
- Functional: Software product being developed for its diverse functionality should make use of available sources on the hardware device, such as camera, file system and processor power.

1.2 Objectives

The objective of this Project is to compare available tools for Image Processing, solving Sudoku puzzles and make a prototype of mobile application, which shall scan the image and solve the puzzle or just solve the manually inputted Sudoku puzzle.

1.3 Structure

The current Thesis paper consists of nine chapters. Chapter 2 gives explanation on the terminology used and theoretical background of the Thesis project. Chapter 3 discusses the literature that was helpful to gain some knowledge and possible find theoretical

solutions for the Thesis project. Also, the related works available in the app market such as Play Store are briefly examined. Chapter 4 gives overview of the requirements and its analysis in order to define stakeholders, use cases and etc., which are considered to be the fundamental elements of the Thesis problem definition. Next is Chapter 5, which proposes the theoretical solutions for the defined Thesis problem and the different technology options are analysed. Following Chapter is Chapter 6, where the previous findings are put together and implemented in Chapter 7. Chapter 8 gives detailed view on the evaluation of results and cross check with previously defined requirements. Results and conclusions are examined on the last Chapter.

2 Background

This chapter starts with definition of the terminology used. Afterwards, the necessary background information for Sudoku, digital cameras and mathematical equations are provided. Additionally, the reader gets familiarized with the classification field and image processing algorithms.

2.1 Definitions

Product In the scope of this Bachelor Thesis, the product will be referred as a Mobile Application that is being developed.

Sudoku The type of logical puzzle that is the concern of this Thesis problem. Sudoku is a grid of custom sizes, but this Thesis solution will specifically focus on a grid of size 9x9 cells, which comprises of 81 cells and usually in Sudoku puzzle there are some cells already pre-filled. Hence, the task is to find the suitable numbers for the rest of not pre-filled Sudoku grid cells.

User The person who is operating the mobile application.

Detection The task of finding the position of desired object in an image, in our case Sudoku board and its numbers.

Image Processing The steps required for transforming the image into desirable state. The desirable state for this Thesis is image processed picture, which is inputted for the classification model and the output of classification model is the prediction, which is final result.

Classification Classification [1] is a technique for categorizing given data into a number of classes. The main problem of classification is to find the category/class to which input data fall into.

Backtracking algorithm Backtracking [2] is an algorithmic technique for the recursive problem solution by building final solution incrementally, step by step, removing those solutions, which failed to satisfy the constraints of the problem while traversing the search tree.

Considering the current Thesis problem, we try filling digits one by one. Whenever the wrong digit is found which does not lead to a correct solution, this digit is removed (backtracked) and next digit is tried. This is better than naive approach, where all possible combinations of digits are generated and then every combination is tried one by one) as it drops a set of permutations whenever it backtracks.

2.2 Digital Camera

2.2.1 Digital image

Digital images [3] are the representation of 2 dimensional numbers, which appear in a raster or vector type. The difference between raster and vector image type is that in vector type the image resolution is not fixed and raster type the resolution is fixed. When the vector image is scaled to any size, the quality of image will stay the same. Owing to the reason that most of the digital cameras make raster images, the vector images will not be considered in this Thesis paper. Raster images consist of finite set of numbers, which are referred as picture elements or in short pixels. Effectively, pixels are the smallest individual elements of an image. The raster image consists of a matrix with fixed number of rows and columns of pixels. The centre of pixel coordinate system [4] of image is located at row 0 and column 0, which corresponds to the top left corner of an

[1] https://analyticsindiamag.com/7-types-classification-algorithms/#:~:text=Classification%20model%3A%20A%20classification%20model,of%20a%20phenomenon%20being%20observed, (last visited: 12.07.2020)
[2] https://www.geeksforgeeks.org/backtracking-algorithms/#:~:text=Backtracking%20is%20an%20algorithmic%2Dtechnique,reaching%20any%20level%20of%20the, (last visited: 12.07.2020)
[3] https://en.wikipedia.org/wiki/Digital_image , (last visited: 12.07.2020)
[4] https://docs.cornerstonejs.org/concepts/pixel-coordinate-system.html, (last visited: 12.07.2020)

image. Hence, the image resolution of 30x20 indicates that image has width of 30 pixels and height of 20 pixels.

2.2.2 Colour space

Colour space [5] is a particular representation of colours, which enables engineers to reproduce colours in both analog and digital world. Basically, the colour space describes mathematical model, which can be in the form of tuples of numbers, e.g. triples in red, green and blue (RGB) format. The following paragraphs will go through the various color spaces that are relevant for the development of this Thesis work.

Binary In binary image [3] the pixels can have either black or white colour, which lead to the conclusion that each pixel is stored as a single bit, either 0 or 1.

Gray-scale In gray-scale image [6], the pixels are the representation of grey colour with varying intensity. The difference of grey-scale image from binary is that in grey scale, all of the pixels are the different variations of grey colour shades.

HSV Hue, Saturation and Value (HSV) [7] (Figure 2.1) is an alternative representations of the RGB colour model. The objective of HSV colour space was to align more closely with the human colour perception, while customizing the colour for own purposes. HSV describes image colours through hue component, which is expressed as a number between 0 and 360 degrees, the shade of colour via saturation, which spans from 0 to 100 percentage and the brightness using value component, which ranges from 0 to 100 percentage.

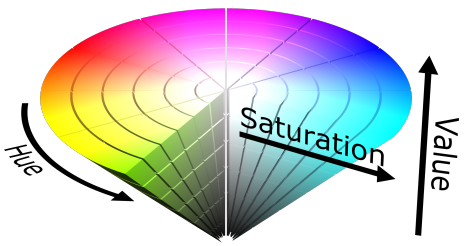


Figure 2.1: HSV color space[5]

RGB Occasionally referred as "truecolor" image [4], RGB image (Figure 2.2) pixel consists of the combination of three colours, which are red, green and blue and the rest of image colours are derived from the above-mentioned RGB colours. Each colour component is very sensitive to the changes in a sense that the colour tone, saturation and brightness alters simultaneously.

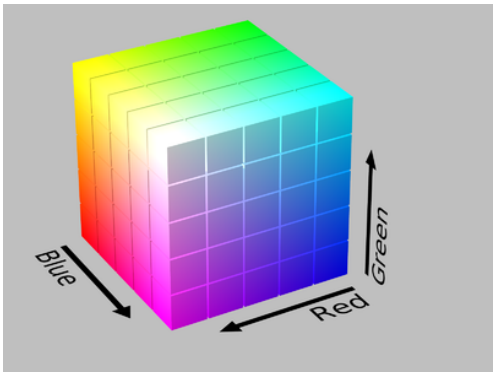


Figure 2.2: RGB color space[6]

2.3 Image processing

IP (Image Processing) [7] field offers different methodologies to enhance or alter the image appearance in order to extract the meaningful information and in our case Image Processing is crucial in order to transform the input image into the one, which our classification model expects. For every basic IP implementation, one can find 3 core steps:

1. Image acquisition via various image import tools
2. Pre-process and manipulate image
3. Output the altered image

In general, there are 2 types of IP namely analogue and digital. In analogue IP the hard copies like printouts, photographs are the input for IP. In contrary, digital IP serves for manipulation of digital images by using computers.

[5] https://commons.wikimedia.org/wiki/File:HSV_color_solid_cone.png, (last visited: 12.07.2020)
[6] https://en.wikipedia.org/wiki/HSL_and_HSV, (last visited: 12.07.2020)
[7] https://sisu.ut.ee/imageprocessing/book/1, (last visited: 12.07.2020)

2.3.1 Thresholding

Thresholding [8] is the basic method for image segmentation. Basically, thresholding converts colour or grey scale image into a binary image, i.e., one that is simply black and white. There are different thresholding methods, for our purposes we are going to explore adaptive thresholding. The idea of adaptive thresholding is to apply varying image thresholding, which differs for each small areas of image.

2.3.2 Blurring

The objective of using blurring or occasionally called smoothing [9] is to average out considerable changes of pixel intensities. As a result, the noise is reduced from image. In other words, blurring is applying low-pass filter to an image, which means reducing noise from an image, but the drawback is that typically the majority of the image is not left flawless. The following are the possible blurring methods that have been considered in the development of this project:

Mean filter Also known as box filter and average filter [10], mean filter is based on mathematical geometric mean, which is shown below:

$$G(x,y) = round(\frac{1}{mn} * (\sum_{i=0}^{n}(\sum_{g=0}^{m} S(i,j))))$$

Figure 2.3: Mathematical representation of mean filter

S(i,j) is input image, ***i, j*** means i,j is an element (pixels) of the set S (input image), and the filter mask is **mn** pixels. Every pixel of the output image at point (x,y) results from the sum of the pixels within the mathematical mean mask divided by 1/mn.
As an example, if we take a mask size of 3 by 3, the pixel x,y in the output image will be the sum of S(x,y) pixel and the rest of 8 surrounding pixels divided by 1/9.

[8] https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm#:~:text=Adaptive%20thresholding%20is%20the%20method,()%20of%20the%20Imgproc%20class., (last visited: 12.07.2020)
[9] https://datacarpentry.org/image-processing/06-blurring/#:~:text=The%20effect%20is%20to%20average,pass%20filter%20to%20an%20image., (last visited: 12.07.2020)

The table with pixel x,y at the centre will give the following result:

4	2	3
1	2	3
7	3	5

Table 2.1: Mean filter example

Gives the result of: $\boldsymbol{round((4+2+3+1+2+3+7+3+5)/\frac{1}{9}) = 3}$

Gaussian blur Known as Gaussian smoothing [10], Gaussian blur is used to reduce image noise and detail. It is widely used technique in image pre-processing in order to accent on more general image structures. In digital image processing, it is common to use binomial filter [9], which is approximation of Gaussian filter. Mathematically speaking, Gaussian blur is convolution of image with Gaussian function. In other words, Gaussian blur acts as a low pass filter in a sense that high frequency components of image are reduced.

2.4 Classification

In general term, classification [15] is a process of categorization, in which the classification objects are recognized and differentiated according to the previously identified specific object parameters.

In Machine Learning (ML), the classification is a type of supervised learning, where the target classes are specified. In other words, all supplied input, which is data element belong to specific class and these classes are referred as target classes.

There are various types of classification algorithms and two general algorithm Models that are linear and nonlinear. Linear models encapsulate the following:

- logistic regression is used for binary classification problems
- support vector machines find optimal differentiating line between different target classes

Nonlinear models encapsulate the following:

[10] https://en.wikipedia.org/wiki/Gaussian_blur (last visited: 12.07.2020)

- K-nearest Neighbours (KNN) identifies characteristic features from the data sets position on plane
- Kernel Support Vector Machines (SVM) finds relations in given datasets
- Naive Bayes classifies data with no relation to the presence of another pattern
- Decision Tree splits data based on different "tree" conditions
- Random Forest classification finds an average from the outputs of tree leaves

Classification has broad spectre of usage in various fields e.g. in finding whether email is spam or not, to identify if kid will pass or fail examination and etc. In our case, classification is used in identifying the numbers contained inside the extracted cells of Sudoku puzzle after Image Processing step.

Figure 2.4 shows the classification example in form of chart that indicates the classification of Iris flower dataset into three sub-species indicated by 0, 1 and 2 numbers.

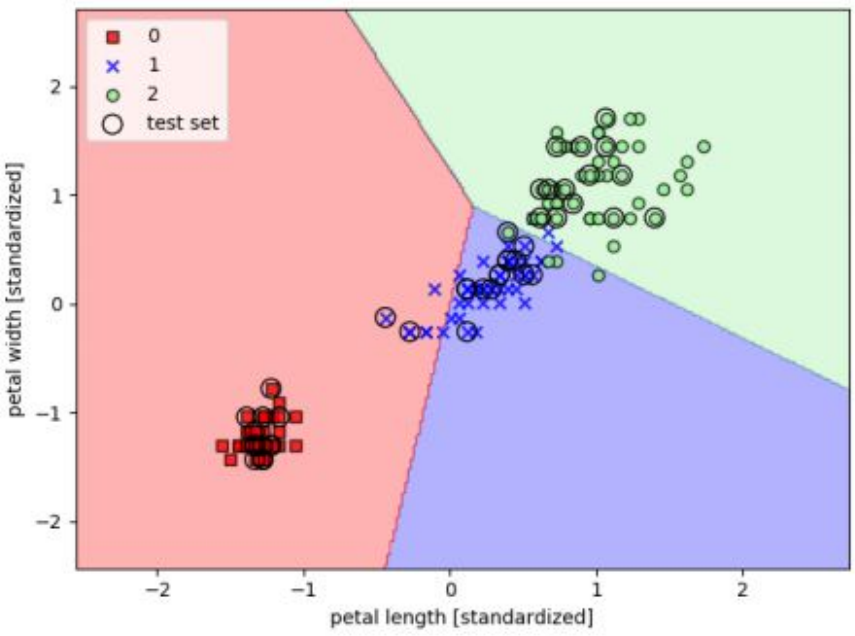


Figure 2.4: Classification example

2.5 Android platform

In this section, the core Android concepts such as manifest, activities, resources for layout, values by xml files and etc. will be discussed.

2.5.1 Manifest

Manifest file [11] is one of the core files located at the root of the project source set. The manifest file provides information to Android build tools, Android operating system and Google Play. In every manifest file, one must include the following:

- The package name of the app and in most of the cases it is the same as source code name space. The Android build tools use the package name in order to find the location of code entities while building the project.
- The core components of the app, which encapsulate all activities, services, broadcast receivers and content providers. Also, the supported device configurations, intent filters can be included.
- The permissions to access protected parts of the system that the app requires. Also, different app interaction permissions can be declared in the manifest file.
- The required hardware and software features for the app operation are declared here and this indicates which devices can install the app from Google Play.

In case Android Studio is used to build the app, the manifest file with all essential manifest components is created.

2.5.2 Activities

The Activity class [12] is important component of the Android application, because when the application starts, Android system executes the content of the main activity class. Also, the interaction between different apps takes place with invoking an activity in the other app. Hence, activity acts as an entry point for an users' interaction with the

[11] https://developer.android.com/guide/topics/manifest/manifest-intro, (accessed 10.07.2020)
[12] https://developer.android.com/guide/components/activities/intro-activities, (accessed 10.07.2020)

app. Any activity is implemented as a subclass of the Activity class. Furthermore, each activity provides the window in which the apps draws its UI. The pre-requisite of using activities is to include the activity name in the app's manifest file.

2.5.3 Resources

Resources [13] are the complimentary files, which app uses e.g. bitmaps, layout definitions, static strings used in layouts, images, animation instructions etc.

Resources folder enables the developer to make the code maintenance easier by putting all of the static code into single folder, where all corresponding app components are structured and separated in different sub folders. Also, it gives the chance to change the app layout and its components depending on the configuration. For example, if the different app layouts are supplied for different screen sizes, then the corresponding layout for the required screen size of mobile device will be loaded and shown to the user.

2.5.4 Layouts

Layout [14] describes the UI elements present in the app and their corresponding structure. All elements are built using View object and the group of View objects is called ViewGroup. View shows the visible elements that user can see and interact with. ViewGroup is an invisible container, which consists of Views and possibly other ViewGroups as shown in Figure 2.5.

2.5.5 File system

File system [15] is an interface and factory for objects to access files in the mobile device file system. It is required to define the permission requests for reading or writing file in manifest file.

[13] https://developer.android.com/guide/topics/resources/providing-resources, (accessed 10.07.2020)
[14] https://developer.android.com/guide/topics/ui/declaring-layout, (accessed 10.07.2020)
[15] https://developer.android.com/reference/java/nio/file/FileSystem, (accessed 10.07.2020)

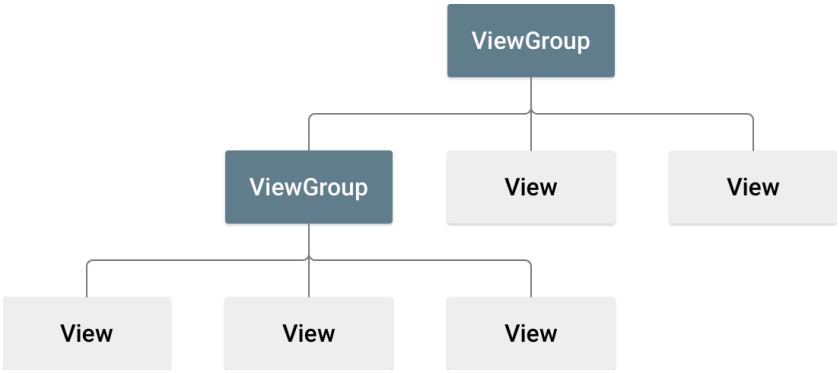


Figure 2.5: Illustration of a view hierarchy, which defines a UI layout

2.5.6 Camera

In Android framework, image and video capturing is supported through android.hardware.camera2 API or camera Intent [16]. It is important that before making use of camera on Android device, the camera permissions need to be appropriately declared inside manifest file. Also, declaring the Camera permissions inside manifest file will prevent the application to be installed to devices that do not support camera features on the device.

[16] https://developer.android.com/guide/topics/media/camera, (accessed 10.07.2020)

3 Related work

In this chapter the resources, which were found on the Internet are briefly examined and the ones that I found interesting are discussed in more details. Some explored techniques have been helpful in order to understand the efficiency and also the drawbacks of certain techniques. The studies conducted show that the Image Processing techniques are crucial in order to identify the number located inside Sudoku grid cells. Although, there are certain image parameters such as image noise which dramatically affect the success of Sudoku grid cell number identification.

3.1 Literature research

There are several resources that give suggestions and strategies in order to solve the Sudoku problem. The one which I was recommended by Professor Rauscher-Scheibe is the magic square, which is straightforward and simple strategy.
The principle idea of a magic square [1] is that the row, column and main diagonals all add up to the same number. But, in this Thesis more simple and primitive approach has been used, which is backtracking algorithm.

The work [14] indicates the similar problem of identifying and solving Sudoku puzzle. The above-mentioned work specifies the number of image processing techniques such as thresholding, extraction of outer box, finding connected lines and many more. For number identification, Object Character Recognition (OCR) has been applied, which is effective only when the extracted numbers don't contain noise, or it is minimal amount. Hence, it is important, to thoroughly filter the image in order to get sharp and lucid image for OCR.

[1] https://plus.maths.org/content/anything-square-magic-squares-sudoku, (last visited: 12.07.2020)

Another work [18] also indicates a vision-based Sudoku solver, which as it is said is capable of solving Sudoku directly from an image captured by any digital camera. After image pre-processing, the area calculation techniques are applied in order to recognize the enclosing box of the puzzle. Afterwards, a virtual grid is drawn to identify the number positions enclosed inside the Sudoku grid cells. As a method for digit recognition, template matching is utilized and in order to solve Sudoku the popular Backtracking algorithm is applied.

The work of [16] describes the development of Android mobile application which automatically scans and solves Sudoku puzzle. The developed system starts with supplying image using built-in camera or the image is loaded from the device media gallery. Afterwards, contour finding algorithm and Hough transform detect Sudoku game board. 81 cells of Sudoku grid are extracted using contour finding algorithm and the numbers located inside extracted cells are identified using OCR. Eventually, Sudoku is solved using Backtracking algorithm and the result is depicted in the form of an image.

3.2 State of the art Android applications

Sudoku scanning and solving Mobile Applications that are available at Play Store have been analyzed and the best practices have been adopted. Market research has helped to shape the future app idea and raw app wireframe.

3.2.1 Sudoku Solver: Solve sudoku from picture or camera

«Sudoku Solver: Solve sudoku from picture or camera»[2] application incorporates the IP and Sudoku solution, which as it can be seen delivers good results, but User Interface (UI) is very simplistic, because it has very basic layout and colour choices are black, white and grey. UI could have been improved by giving more various colours and different shapes to the layout of the screen views, buttons, text and etc. Additionally, for some images the app is crashing. However, the general practices used for this mobile application such as different image acquisition, manual Sudoku grid operations and others have been noted and included in the features list of my future application. It definitely shows the pros and

[2] https://play.google.com/store/apps/details?id=com.companyname.sudokusolvermobile, (last visited: 06.07.2020)

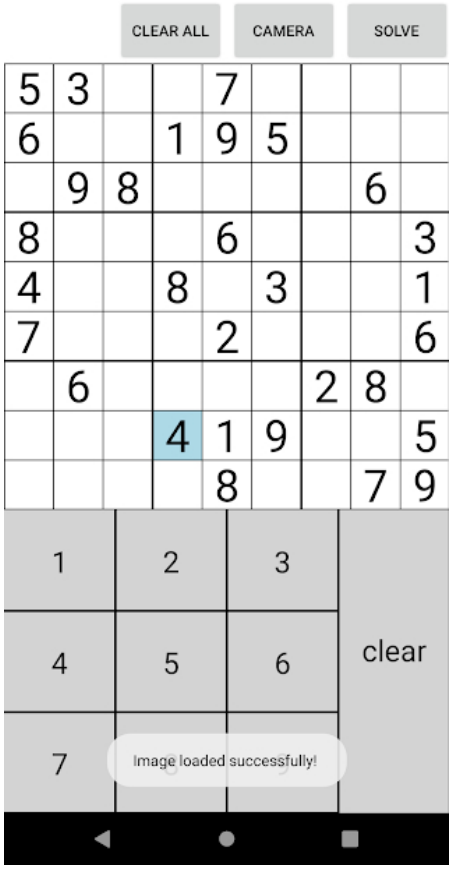
cons of the design simplicity and tech approach used. Advantages of the design simplicity is that it is relatively easy to implement and maintain, but from the disadvantages, in most of the cases the design simplicity is boring for the potential app users.

3.2.2 Sudoku Scan&Solve

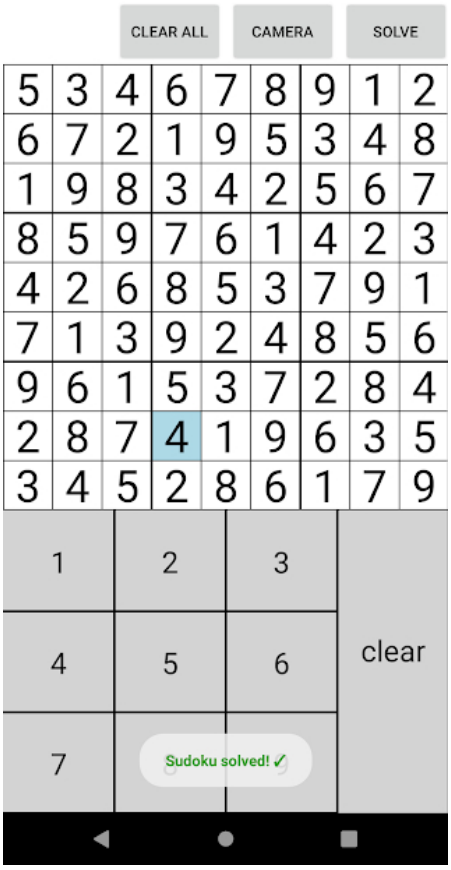
Probably, the most promising result was obtained from «Sudoku Scan&Solve»[3] application. This mobile app makes use of popular mobile IP tool OpenCV. The workflow and user tools are quite intuitive, but the waiting time for Sudoku recognition can get longer than 3 minutes. However, so far this app has given the highest accurate results in identifying the numbers located inside Sudoku. As in most of the Sudoku scanning and solving mobile applications, the permission for camera and device file system is requested in order to take picture and operate with data on device file system.

[3] https://play.google.com/store/apps/details?id=com.stephanwidor.sudokuscanandsolve, (last visited: 06.07.2020)

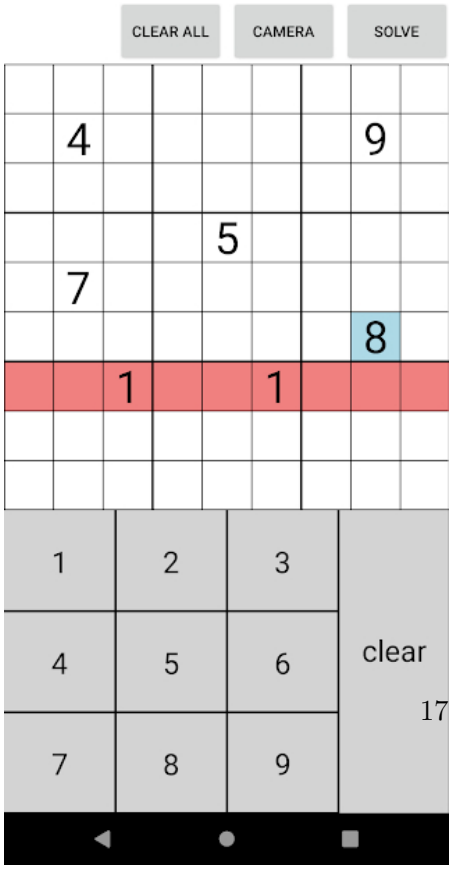
(a) Logo of Sudoku Solver: Solve sudoku from picture or camera



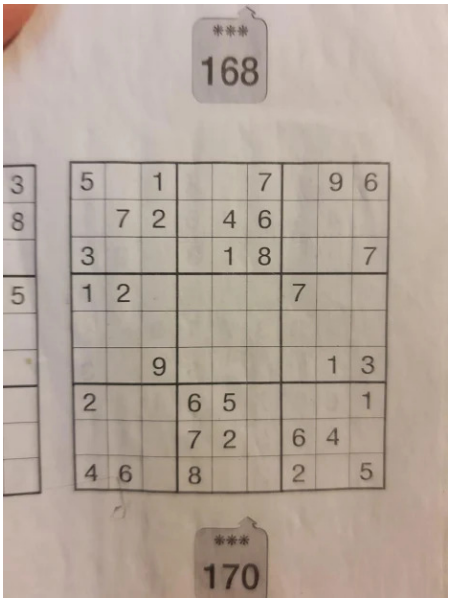
(b) Screenshot of Sudoku Solver: Solve sudoku from picture or camera



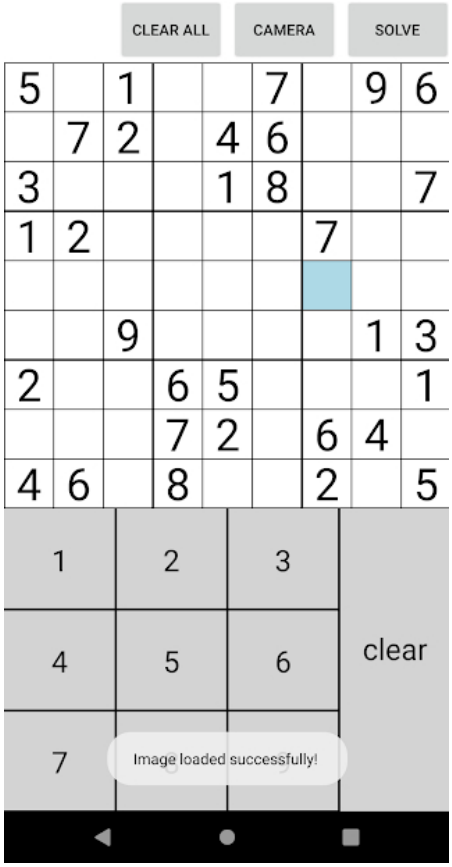
(c) Screenshot of Sudoku Solver: Solve sudoku from picture or camera



(d) Screenshot of Sudoku Solver: Solve sudoku from picture or camera

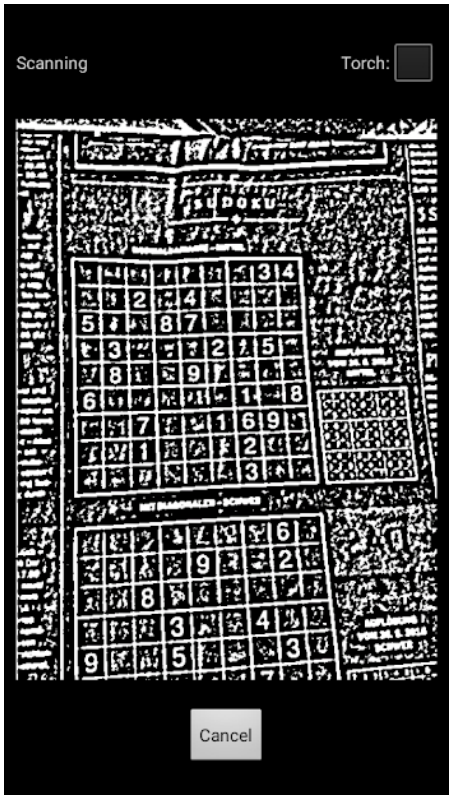


(a) Screenshot of Sudoku Solver: Solve sudoku from picture or camera

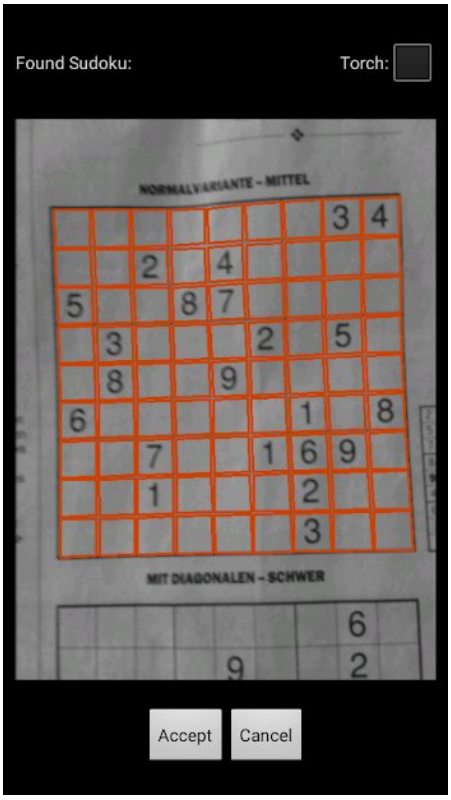


(b) Screenshot of Sudoku Solver: Solve sudoku from picture or camera

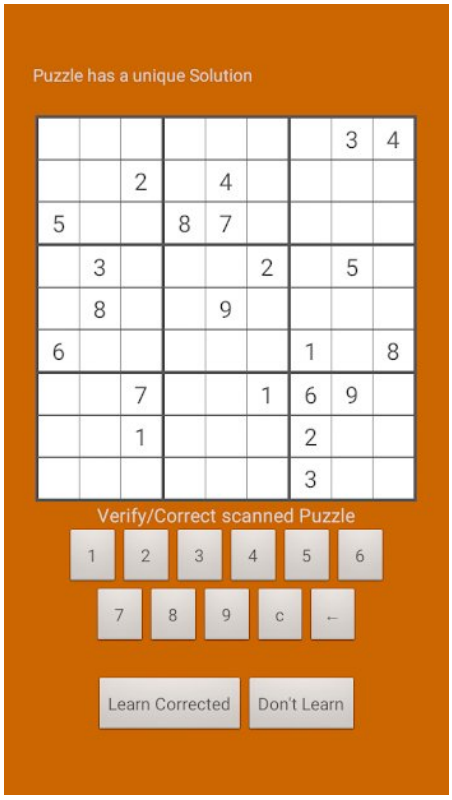
(a) Logo of Sudoku ScanSolve



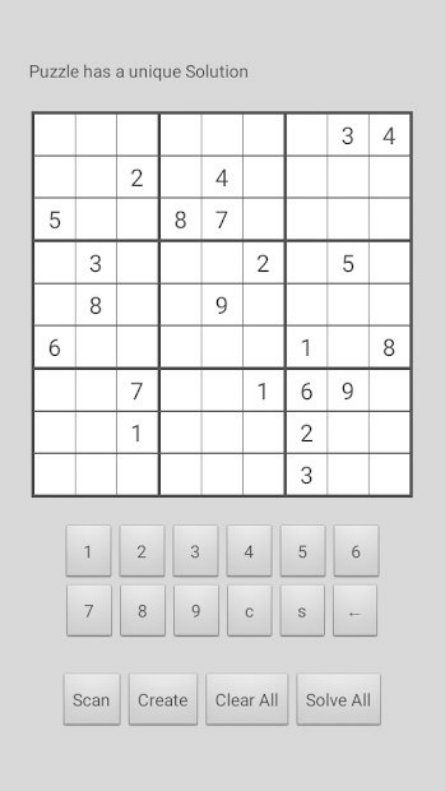
(b) Screenshot of Sudoku ScanSolve



(c) Screenshot of Sudoku ScanSolve

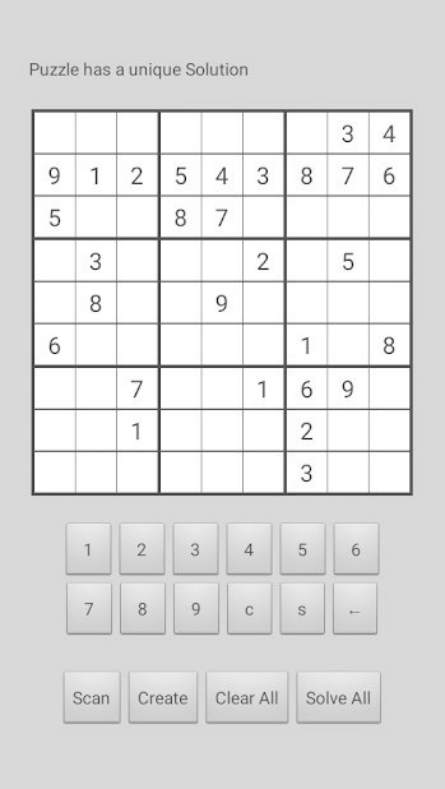


(d) Screenshot of Sudoku ScanSolve



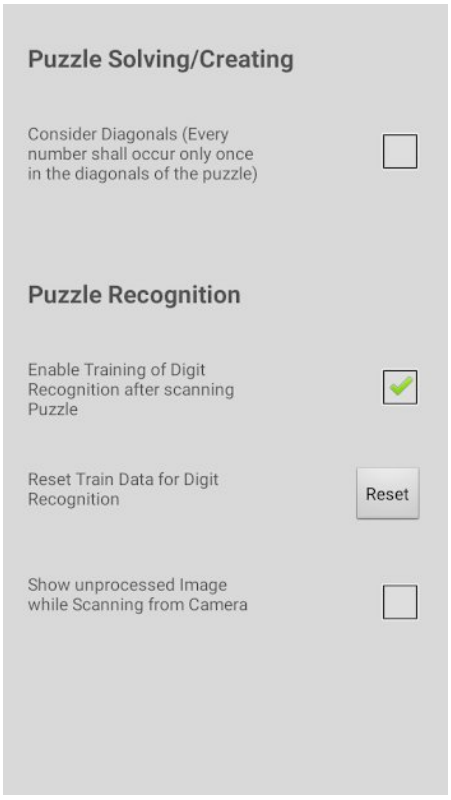
(a) Screenshot of Sudoku ScanSolve

(b) Screenshot of Sudoku ScanSolve

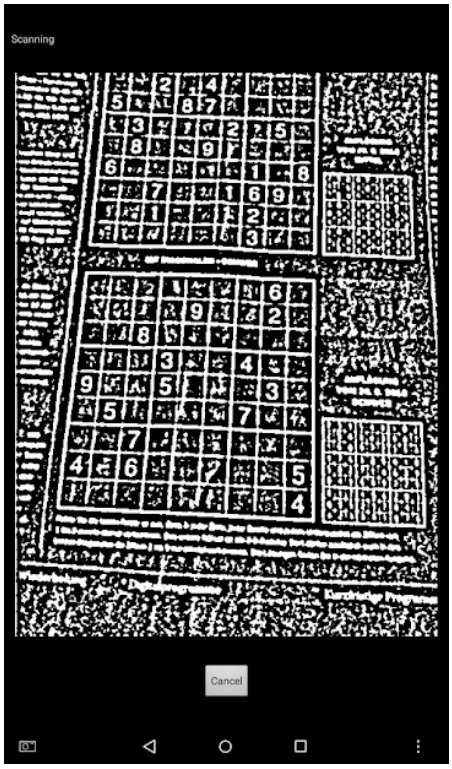


(c) Screenshot of Sudoku ScanSolve

(d) Screenshot of Sudoku ScanSolve

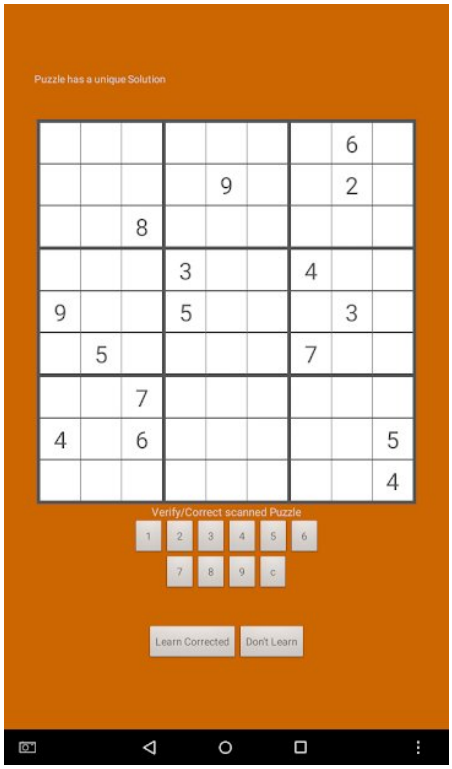


(a) Screenshot of Sudoku ScanSolve

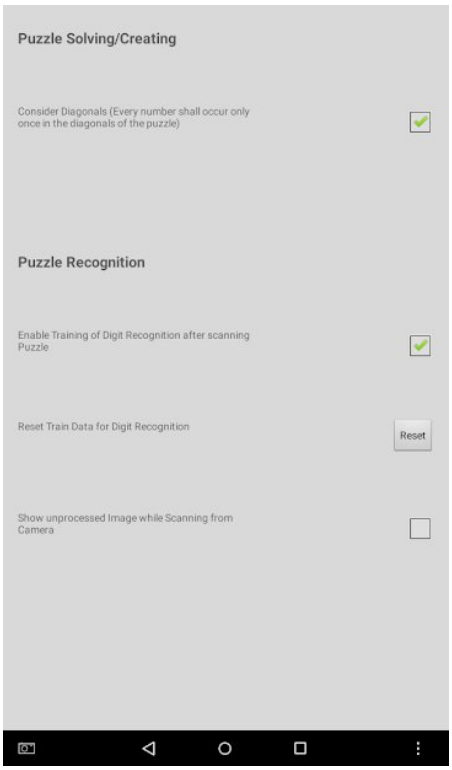


(b) Screenshot of Sudoku ScanSolve

(c) Screenshot of Sudoku ScanSolve



(d) Screenshot of Sudoku ScanSolve



(a) Screenshot of Sudoku ScanSolve

4 Requirements analysis

This chapter gives a detailed view on the requirements analysis, where one can distinguish this concept as the process of explicit definition of the user expectations for a new product being developed or modified. More precisely, the requirements analysis is done in order to obtain the concrete stakeholders, use cases, functional and non-functional requirements. The requirements analysis was approached in the following manner, first the base requirements were predefined by the stakeholder, who is in a matter of fact the Supervisor of this Bachelor Thesis at Hamburg University of Applied Sciences, Professor Marc Hensel. After acquiring the set of base requirements, the stakeholders, use-cases, functional and non-functional requirements were defined and summarized in the following subchapters.

4.1 Process models

This section is a core element of the work being carried out in order to develop the ready product, which in this case is Mobile Application. In this Thesis, the Agile Model has been widely practiced and it is depicted in Figure 4.1.

Agile Model [13] is one of the most widespread software development models and one can find 6 core steps defined for this process model namely Requirements analysis, Design, Implementation, Testing, Deployment and Maintenance, which is gathering the feedback and re-iterating with the feedback as new requirements. As it can be visually seen from Figure 4.1, the general aspect of the Agile model is a circle, where the core operations are being carried out. Agile model process is triggered by Requirements and the output of each iteration is deployment of the product that is being developed during this whole process.

As it has been mentioned, for this Thesis work the Agile Model approach has been used and in the following sentences the detailed explanation is to be given. The concept of

sprints from agile methodology can be related to our meetings with my Supervising Professor and the idea of backlog can be correlated to the plan, which is made after each meeting. The reason why it is not Waterfall model is because software test was conducted concurrently with the development process rather than testing after the full completion of software. Also, the fact that the approach for this Thesis was iterative and incremental, V-Model doesn't qualify as a process model.

For this Thesis work, the initial planning with Supervising Professors triggered the process of requirements analysis, where the stakeholders, use-cases, functional and non-functional requirements were identified. The requirements analysis step lead to the next stage of design and implementation of "Scan&Solve Sudoku" Mobile Application. At this step, the solution is to be tested and deployed into production namely PlayStore. The last step is the review, where the users' feedback is evaluated. Review is significant step in a sense that the new iteration will be triggered according to the summary made from the previous iteration.

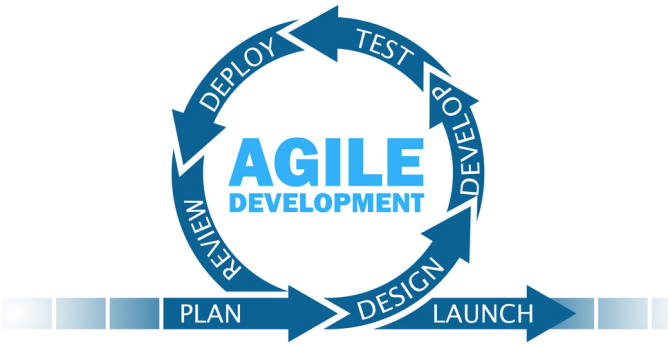


Figure 4.1: Agile model[1]

4.2 Stakeholders

In order to give the general idea of this sub chapter, first the definition of stakeholders is to be given. Stakeholder is a person, group or organization that has interest in a project

[1] https://www.cleart.com/wp-content/uploads/2017/11/what-is-agile-software-development.jpg, (last visited: 12.07.2020)

that they are part of and can either affect or be affected by this project. Stakeholders are summarized in the table below. They can be categorised into internal and external groups.

Number	Stakeholder
1	Supervising Professors
2	App user
3	Developer of the current app
4	Public Society

Table 4.1: Stakeholders

Internal group represented by professors from Hamburg University of Applied Sciences: the supervising professors from whom this idea to use algorithms in order to solve Sudoku for math courses as an instance and in the end, it was proposed to automate this in an app. During the development process, 1st Supervising Professor is taking part in checking the progress and correctness of delivered content for this Thesis. The second Professor will be acting as a referee for the completed Thesis paper and in the end both of the Supervising Professors will give their final grade for this Bachelor Thesis work. Thus, both of the Supervising Professors are interested in the results of this paper acting as a referee and hopefully using the Mobile Application that is being developed for Math and Software Engineering classes. As an instance of potential app user, the students are considered to be interested in this app owing to the fact that this Mobile Application will help them in solving the logical question quizzes. Hence, students, who are an instance of users of this Mobile Application, don't take part in the development process, but they are considered to be the end-users of this Mobile Application. Developer of the current app, named "Scan&Solve Sudoku" on PlayStore, is interested in the success of this Thesis and can directly impact the state of the app. Moreover, developer is a person who is responsible for the maintenance and further development of this Mobile Application.
The external group is represented by public society, which might use the published app that is being developed for this Thesis. The current Mobile Application is uploaded to Play Store and the public society is able to access the app from Play Store. With the user feedback there is a chance that the app will be developed further, and new releases will be published by improving the functionality, fixing the bugs and changing User

Interface in general. As of now, this app is not considered to be commercial, thus no revenue generation is expected.

4.3 Use cases

From the perspective of end users of this mobile application the use cases are defined on the following pages. The essential goal of this Thesis work is to solve the certain type of logic puzzles which is Sudoku. Thus, at first step the Mobile Applications Market has been analysed in order to explore the technologies, approaches and features used. With the corresponding conclusions made, the suitable Sudoku solver algorithm has been found in order to solve the puzzles and the Computer Vision library has been chosen which would perform relatively effective with given resources. It makes sense to give the user the possibility to save the solved questions in order to review it later or pause Sudoku solving and continue later. Hence, this functionality avoids the redundant user action of taking picture again. Considering actors, the Mobile Application user is the operator, who in the matter of fact starts the application, loads the picture and receives the ready solution of Sudoku puzzle. In order to operate to the full-extend the current Mobile Application, user's phone should be equipped with camera and give the access permission to Mobile Device's storage. All above-mentioned information is summarised and depicted in Figure 4.2.

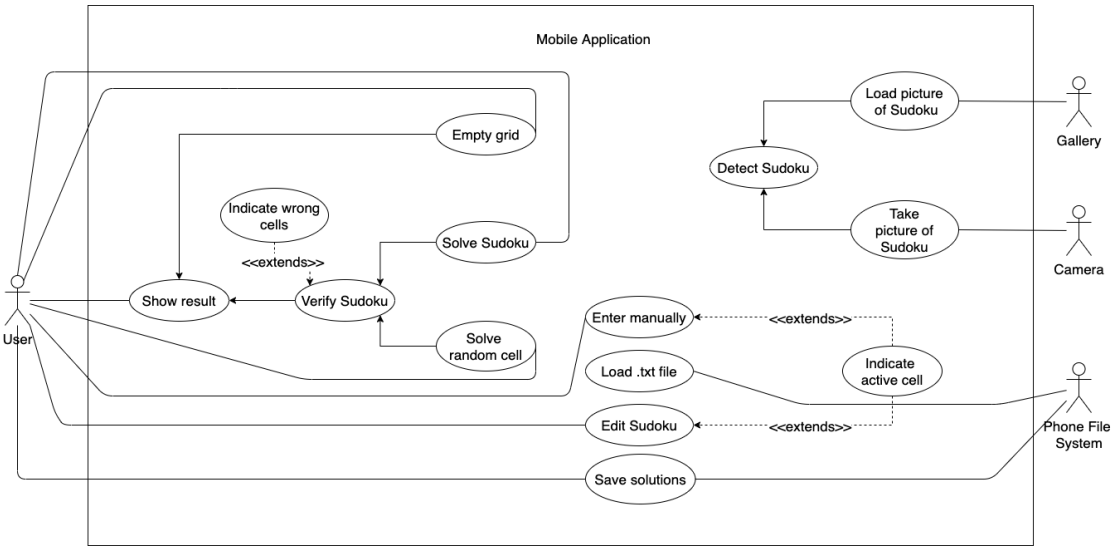


Figure 4.2: Use case diagram of the system

Use case	Load picture of Sudoku
Description	User opens gallery of device and loads picture of Sudoku puzzle
Actors	User
Systems involved	Gallery of device
Trigger	User presses the button
Preconditions	-The read file permissions have been granted -Image of Sudoku puzzle is available in gallery of device
Success end condition	Picture with Sudoku is loaded on the window of application
Failure end condition	-The read file permission are not granted -There is no image of Sudoku in gallery of device -poor quality image
Typical flow	1. User presses the "Load picture" button 2. Picture is loaded on the window of application and used for further processing

Table 4.2: Use case table for "Load picture of Sudoku"

Use case	Take picture of Sudoku
Description	User launches camera of device, takes picture of Sudoku puzzle, afterwards the shot image is loaded to the application
Actors	User
Systems involved	Camera
Trigger	User presses the button
Preconditions	-Camera permission is granted - Sudoku puzzle is present in image
Success end condition	Picture with Sudoku is loaded on the window of application
Failure end condition	-The camera permission is not granted -There is no image of Sudoku in shot image -poor quality image
Typical flow	1. User presses the "Take picture" button 2. Shot image is loaded on the window of application and used for further processing

Table 4.3: Use case table for the "Take picture of Sudoku"

Use case	Detect Sudoku
Description	Detects Sudoku board, its numbers
Actors	Gallery or Camera
Systems involved	Image Processing library
Trigger	User presses the button
Preconditions	Image of Sudoku puzzle is loaded to the application
Success end condition	Sudoku board and its numbers are identified
Failure end condition	-there is no Sudoku on image -poor quality image
Typical flow	1. User presses the "Detect Sudoku" button 2. Numbers are loaded inside of Sudoku board cells

Table 4.4: Use case table for "Detect Sudoku"

Use case	Enter manually
Description	User enters the values manually inside of Sudoku grid
Actors	User
Systems involved	-
Trigger	User presses the cell of Sudoku grid
Preconditions	Sudoku grid is loaded on app screen
Success end condition	Sudoku is successfully modified with manual user input
Failure end condition	Sudoku grid is not loaded on app screen
Typical flow	1. User presses the "Enter manually" button 2. Enters numbers into Sudoku board cells

Table 4.5: Use case table for "Enter manually"

Use case	Solve Sudoku
Description	Gives solution for the inputted Sudoku grid cell numbers
Actors	User
Systems involved	-
Trigger	User presses the button
Preconditions	Sudoku cell numbers are correct
Success end condition	Sudoku is successfully solved
Failure end condition	Sudoku cell numbers are wrong
Typical flow	1. User presses the "Solve" button 2. Sudoku is solved

Table 4.6: Use case table for "Solve Sudoku"

Use case	Verify Sudoku
Description	Sudoku is checked for the wrong cells inside of Sudoku grid
Actors	User
Systems involved	Backtracking algorithm
Trigger	User presses the button
Preconditions	Sudoku cell number is entered
Success end condition	Sudoku is successfully verified
Failure end condition	Sudoku cell numbers are wrong
Typical flow	1. User presses the "Solve" or "Solve cell" button 2. Sudoku is verified

Table 4.7: Use case table for "Verify Sudoku"

Use case	Load .txt file
Description	Saved solutions are displayed to user
Actors	Phone file system
Systems involved	Phone File System
Trigger	User presses the button
Preconditions	-Read access permissions are granted -Previously saved solutions are available at device internal storage
Success end condition	Sudoku is successfully loaded
Failure end condition	-Read access permissions are not granted -The loaded file is not previously saved solution of Sudoku
Typical flow	1. User presses the "Load .txt file" button 2. Sudoku grid and numbers are loaded into the app

Table 4.8: Use case table for "Load .txt file"

Use case	Solve random cell
Description	Random cell of Sudoku grid is solved
Actors	User
Systems involved	Backtracking algorithm
Trigger	User presses the button
Preconditions	Sudoku is successfully verified
Success end condition	Random cell of Sudoku grid is solved
Failure end condition	-Sudoku grid cells verification failed
Typical flow	1. User presses the "Solve cell" button 2. Random cell of Sudoku grid is solved

Table 4.9: Use case table for "Solve random cell"

4.4 Functional requirements

The functional requirements, which are derived from previous sections and obtained from stakeholders are listed in Table 4.2. They are prefixed with letter "F" and will be referred throughout the text.
The first and important functional requirement is the requirement F1, which was received from Supervisor. It indicates that the software solution must solve one type of puzzle, which is Sudoku. This requirement makes the whole logic and complexity easier in a sense that it is not required to solve all types of logic puzzles, but only concentrate on Sudoku type of puzzles.
The requirement F2 imposes extra task by demanding the software solution to have Image Processing implementation in order to recognise the objects from picture, which in our case are Sudoku board, cells and numbers. It's the task, which can't be execute with full accuracy, because of image quality and the speed of completion depends on the processing power of device.
The requirements from F3 through F13 declare functions, which focus on usability and extra features of the product. They indicate extra data handling tasks. Among them,

requirement F4 has considerable impact due to the reason that it will improve the usability of system in a sense that there won't be need to retype the Sudoku puzzle.

Encoded number	Functional requirement
F1.	The software solution must solve Sudoku puzzle
F2.	The software solution must recognise the Sudoku board and numbers located inside of the board from picture
F3.	The user must be able to take picture
F4.	The user must be able to import picture
F5.	The user must be able to load .txt file, which consists of the numbers only
F6.	The user must be able to input manually
F7.	The user must be able to save the solved question
F8.	The user must be able to sort the saved solution according to name, date
F9.	The user must be able to filter the saved solution according to name, date
F10.	The user must be able to get hint on next random cell
F11.	When user presses solve button, the software must check user input for the wrong values entered inside of grid cells and indicate the wrong grid cells
F12.	When clicked on "Solve" button, if the wrong cells are not found, the Software shall give the full correct solution
F13.	When clicked on "Empty" button, all of the grid cells shall be cleared

Table 4.10: Functional requirements

4.5 Non-functional requirements

The peculiarity of non-functional requirements is that it shows the quality attributes of the system. They are listed in Table 4.3 and non-functional requirements are listed with

prefix "NF". NF1 indicates that the there should be certain maximum time limit for the detection of numbers in the image. NF2 indicates the necessity for user acceptance test, it must be verified and be part of acceptance test that the acquisition works but this contains too many parameters at the moment to be addressed at the Bachelor Thesis. NF3 is vital requirement concerning the performance due to its positive input to usability. The lower Android version requirements will impact the user satisfaction with a product in a sense that wide range of users will be covered.
Requirement NF4 specifies the elimination of the need to purchase external hardware, which will undoubtedly have the positive impact on users' experience.
Requirement NF5 states that the main programming language for the implementation of business logic shall be Java.

Encoded number	Non-functional requirement
NF1.	The solution of Sudoku puzzle shall be ready maximum in 1.2 seconds, this maximum time is given, because the Software performance differs slightly on different hardware.
NF2.	The user acceptance test shall be defined for the speed of the identification of objects on picture, because each mobile hardware device has varying performance parameters.
NF3.	This software solution, which is based on Android platform must support the minimum sdk version for Image Processing library and above API levels.
NF4.	If software solution oriented on mobile devices, no external device shall be required.
NF5.	Programming language shall be Java.

Table 4.11: Non-functional requirements

5 Conception

This chapter concentrates on making solution concepts that meet the requirements of the previous chapter. First, the assumptions are made, then the theoretical, hardware and software solutions are developed.

5.1 Assumptions

Distance between Sudoku and camera Non-functional requirements list (Table 4.11) does not contain any requirement for considering distance between Sudoku and camera. Thus, it is assumed that the camera image covers the full Sudoku grid.

Orientation The device camera can be positioned in any way owing to the reason that image will be adapted and sudoku grid has 9x9 cells. However, if the image is upside down, the number recognition will give wrong results.

Stability The camera must be kept with no visible motion blurring.

Lightening The presence of light in image is acceptable as long as the Sudoku numbers are easily distinguishable.

Sudoku is main element Sudoku puzzle must be the main element of the image.

Nothing that overlaps the Sudoku puzzle Sudoku puzzle image shall not have any overlapping image elements.

5.2 Theoretical Solution and Failure Modes

Theoretically, identifying the numbers is widely practiced problem and there are existing libraries that solve this problem and the corresponding computer vision

libraries are discussed in Chapter 5.4.1. However, each library has its own strengths such as accurate results, wide variety of tools and weaknesses in terms of longer operation times, little community support and considerable processing load. However, for this problem it is required to identify the Sudoku grid and numbers contained inside of the grid. Theoretically in order to distinguish the Sudoku grid, the set of preprocessing steps such as geometric transformations, character recognition and other consequent steps are required. Thus, it was decided to make use of character recognition, detecting lines, thresholding, dilation and other available image processing tools. All of these tools are provided by popular and widely used OpenCV image processing library. However, there are several failure modes, which are summarized in Table 5.1, then analyzed and tackled in the following paragraphs. It should be noted that the below-mentioned failure modes are mostly typical to the majority of Image Processing libraries.

Id	Description
FM1	Sudoku grid lines are not straight in the image, thus cells are extracted wrongly
FM2	Not well-distinguishable lines and Sudoku numbers
FM3	Supplied image is upside down, which results in wrongly identified numbers

Table 5.1: Possible failure modes

Sudoku grid lines are not straight in the image, thus cells are extracted wrongly (FM1) It is required that the Sudoku grid lines are straight, because the current image processing implementation will divide the puzzle in 81 cells and if lines are not straight, then some of the extracted cells will be distorted and the number will appear not in full width or height.

Not well-distinguishable lines and Sudoku numbers (FM2) The lines and Sudoku numbers are not well-printed or good distinguishable from the picture.

Supplied image is upside down, which results in wrongly identified numbers (FM3) It is important for the numbers in image to be located in a correct manner, so

that the Classification model can identify the number correctly.

5.3 Hardware

As it has been mentioned in early Chapters, the hardware should be the Android mobile device, but let us have a look at alternatives and in a fact, there are 2 potential major options for mobile device: single board computers and mobile devices. These options are discussed in the following sub-sections.

5.3.1 Device selection

Above discussed single board computer (SBC) [1] is a complete computer built on a single circuit board, with microprocessors, memory, input/output (I/O) and other features required for a functional computer. Examples of such computers are Arduino Uno and Raspberry Pi.

Arduino Uno Arduino [2] is a microcontroller board that has no operating system and can only run one program at a time, thus it's not designed for multitasking and is best used for simple repetitive tasks. It is easy to setup and start prototyping. On the contrary, it is not powerful enough for resource demanding Computer Vision computations as in our case.

Raspberry Pi 4 Raspberry Pi is fully functional computer with its dedicated processor, memory and is capable of running operating systems such as Linux, Windows 10 IoT, Android and Firefox OS. Another advantage is that it is capable of multitasking in a sense that it can run multiple programs at the same time and is faster than Arduino with 1.5GHz CPU clock speed [3] compared to 16MHz [4]. The disadvantage is that the users will have to purchase an extra device in order to use the product being developed for this Project. Additionally, it will cause the time-consuming stage for setting up the development environment: operating system,

[1] https://en.wikipedia.org/wiki/Single-board_computer, (accessed: 12.07.2020)
[2] https://learn.adafruit.com/multi-tasking-the-arduino-part-1/overview, (accessed: 12.07.2020)
[3] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, (accessed:19.06.2020)
[4] https://store.arduino.cc/arduino-uno-rev3, (accessed: 31.05.2020)

external cameras, drivers, display etc.

Smartphones Android and iOS phones are the most wide-spread mobile platforms and their capabilities are increasing with each new release of their platform devices. General package of the smartphones includes the following built-in cameras, number of sensors, powerful processors, easy to use development environment with detailed documentation and big user community. Above-mentioned aspects clearly indicate that smartphones are the best fit for the purposes of this Project.

Aspect	Mobile	Raspberry Pi 4	Arduino Uno
Board types	Motherboard [5]	Series of single board computer	Single board computer
Multitasking	Positive	Positive	Negative
Quick prototyping	Negative	Positive	Positive
Core package includes	Built-in cameras, number of sensors, powerful processors, easy to use development environment with detailed documentation	Cortex-a72 (arm v8) processor, various RAM options starting from 2GB up to 8GB, HDMI, USB, Ethernet ports, OpenGL ES 3.0 graphics card and etc. [6]	ATmega328P microcontroller, 32KB Flash memory, 2KB SRAM, 16MHZ of clock speed and etc.
User community amount	3.5 billions (2020 year) [7]	3 million (2020) [8]	Not available

Table 5.2: Non-functional requirements

To sum up, the smartphones are the choice for this Project due to the affordability, user-friendliness, computing power and ease of development. Moreover, the users will not have to purchase extra hardware in order to use this product, which improves the

[5] https://www.quora.com/What-type-of-motherboard-does-a-smartphone-have-How-does-it-differ-compared-to-a-computer-motherboard, (accessed:12.07.2020)
[6] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, (accessed:19.06.2020)
[7] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, (accessed: 12.07.2020)
[8] https://www.raspberrypi.org/blog/8gb-raspberry-pi-4-on-sale-now-at-75/#:~:text=Raspberry%20Pi%204%20is%20almost,variant%20from%20%2445%20to%20%2435., (accessed: 12.07.2020)

user experience and success of this product.

5.3.2 Platform selection

As it had been mentioned, there are 2 market leaders in the phone sector and the Mobile Applications developers should make a choice between those 2 popular platforms. Luckily, in recent years the tech companies are bringing innovation in the sphere of Mobile Applications development in terms of new programming languages and frameworks, which help to eliminate the choice problem between Mobile Applications development platforms. From my perspective, the most promising cross-platform frameworks are Flutter from Google and ReactNative from Facebook. The benefit of such development is that there will be one code base, which would run on different platforms. Downside of hybrid platforms [12] is that it does not go well with performance centric, graphics-oriented apps. It might still worth testing, but due to the time-constraints it is not feasible. Therefore, the decision has to be made between Android and iOS platforms.

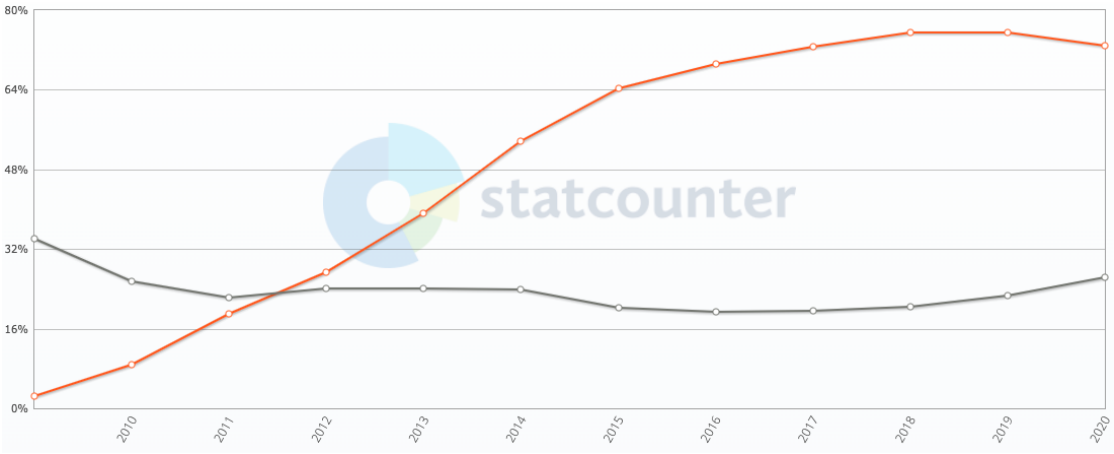


Figure 5.1: Android and iOS market share in the world, indicated with red and black lines respectively[9]

The extensive research on each single aspect of these platforms will not be conducted (but interested reader can find out more [17]), the focus will made on their market

[9] https://gs.statcounter.com/os-market-share/mobile/worldwide/#yearly-2009-2020, (last visited: 07.07.2020)
[10] https://gs.statcounter.com/os-market-share/mobile/germany/#yearly-2009-2020, (last visited: 07.07.2020)

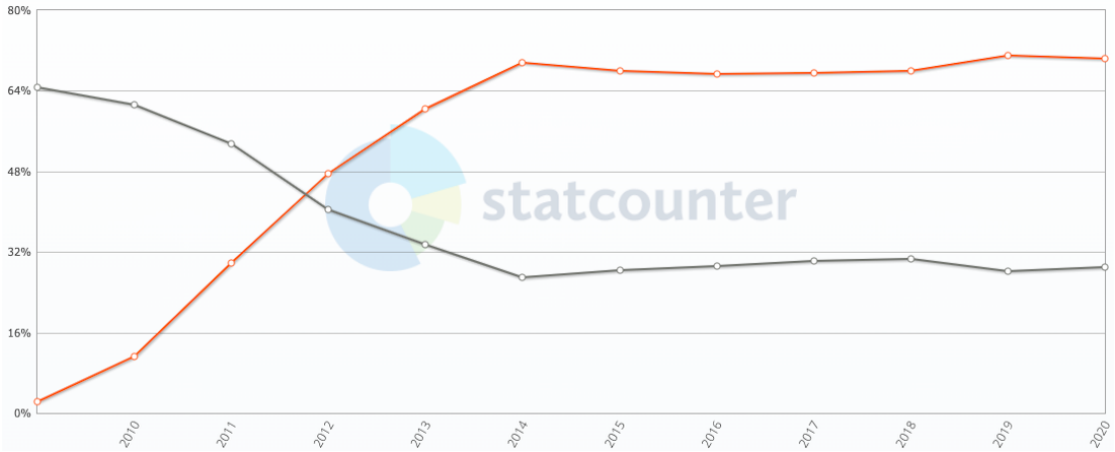


Figure 5.2: Android and iOS market share in Germany, indicated with red and black lines respectively[10]

share. As it can be seen from Figure 5.1, the number of devices on Android platform more than double number of devices on iOS platform. Coming to the mobile device users in Germany, it is vivid from the Figure 5.2 that the same proportion is applicable as the Android and iOS distribution in the world. Therefore, the product will be based on Android platform.

5.4 Software

This section elaborates on the available Computer Vision libraries. Also, different aspects and the development approaches will be defined.

5.4.1 Computer Vision Library Selection

Over the last years, tools for Computer Vision have gained noticeable progress in terms of new libraries and big tech companies such as Google, Microsoft and others being involved. Below, some of the popular libraries will be discussed.

Tensorflow Tensorflow [11] is open-source powerful library for Machine Learning produced by Google. It provides all of the necessary tools in order to create neural networks, train them and get the output model, which can be used for classification,

[11] https://www.tensorflow.org/, (accessed: 02.06.2020)

prediction and other Machine Learning use-cases [12]. Also, Tensorflow can be used for Computer Vision purposes and particularly for our case, it can be used for classification of numbers located inside of Sudoku board cell. However, one needs to have knowledge and experience with Tensorflow in order to create the neural networks, where firstly the necessary dataset needs to be gathered, pre-processed, fed into the custom configured neural network and trained. After training, one shall have small dataset, where the trained model could be tested. Moreover, there should be enough processing power in order to make use of Tensorflow library features. Although, there are already some pre-trained Tensorflow models, one needs to be familiar with Tensorflow library and be able to operate with the model.

OpenCV OpenCV (Open Source Computer Vision) [1] is widely known open-source library for Computer Vision and machine learning. These features enable the developers to apply OpenCV for processing the images and video frames, detection of certain patterns, identification of required elements present in the photo or video frame, classification of objects and etc. Moreover, this library can be used in the variety of programming languages such as Java, C++, Python and others. Above-mentioned aspects make OpenCV library the best candidate for the purposes of this Project.

Matlab Matlab is widely used tool for image processing, data analyses, research, data visualisation and etc. It has own user-friendly Integrated Development Environment (IDE), which allows quick prototyping, error-tracing, debug and others. Basically, Matlab was built for prototyping and research purposes. However, Matlab license is considerably expensive and it is slow during execution time.

To conclude, OpenCV library is chosen for this Project, because it was better in meeting the requirements.

5.4.2 Tools

This subsection will give details about used technology stack, in particular Integrated Development Environment (IDE), programming language, version control system used for Software development.

[12] https://www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/#:~:text=It%20is%20an%20open%20source,%2C%20Discovering%2C%20Prediction%20and%20Creation., (accessed: 02.06.2020)

Android Studio

Android studio [8] is IDE used in developing Mobile Applications on Android platform and it is available for all most popular Operating Systems as Windows, macOS and Linux. Android Studio features Gradle-based build support, Android-specific refactoring, Proguard integration and many others.

Java

Java programming language was preferred to Kotlin, which is another available Android Applications programming language because I am most experienced with Java and OpenCV doesn't support Kotlin.

Git

Git is most popular [13] and free Version Control System (VCS) available in the software market. I have used Git in order to manage the different versions of Software being developed in the current Project.

Gradle

Gradle [2] is JVM-based build tool, which automates the compilation, test and release processes of the Software being developed. Also, gradle is helpful in dependency management, making different build versions of the software and etc. For this Project, gradle has been widely used for building the Android project.

[13] https://hackernoon.com/top-10-version-control-systems-4d314cf7adea, (last visited: 07.07.2020)

6 Design

This chapter will discuss the design details in form of UML diagrams, such as state machine diagrams, activity diagrams and it will give insights into the graphical design of the Mobile Application.

6.1 UML Diagrams

Unified Modelling Language (UML) is used in the following sections in order to describe the system model. These diagrams help in the development of final product.

6.1.1 State machine diagram

As it can be seen from Figure 6.1, there are 4 states for the Sudoku grid and its numbers detection from an image, which is supplied by the user via loading image or taking picture. The activity starts with Idle state, where it waits for Take image or Load image button to be pressed. Afterwards, the selected image is displayed on user's device screen. At this step, if user presses Scan Sudoku button, then the image is processed using Image Processing (IP) library. Next step is to display the detected numbers in new activity, where Sudoku grid is drawn and detected numbers are displayed inside of the grid.

6.1.2 Activity diagram

Activity diagram for use case "Detect Sudoku" (Table 4.2) is shown in Figure 6.2. As illustrated, activity is triggered with a press of "Take image" or "Load image" button, next activity depends on the user choice, if "Take image" button is pressed, then Camera is launched in order to take photo, else if "Load image" button is pressed, the image, which is selected by user will be loaded. Latest possible case is going back to Main window by pressing "Back" button or "Main window" button. As a result, when image is successfully supplied, then it is displayed on device screen. The further action depends on the user choice, if "Scan Sudoku" button is pressed, then the image is

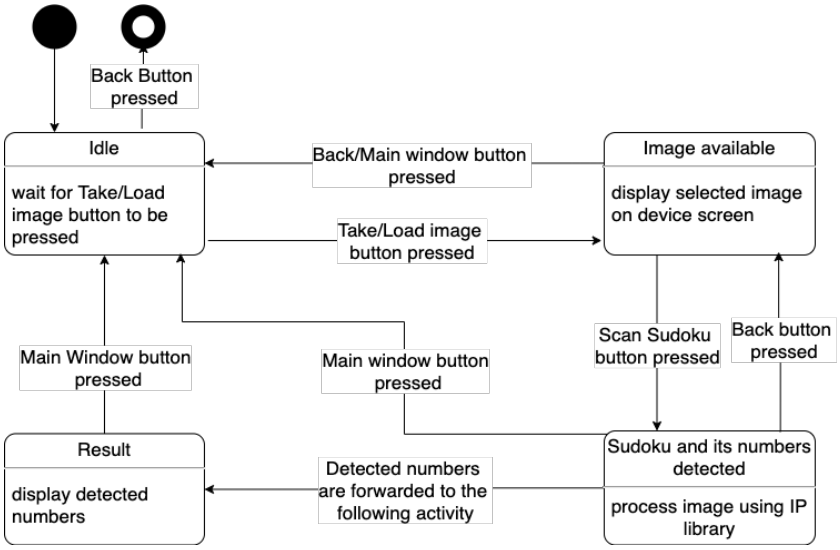


Figure 6.1: State diagram for the detection of Sudoku grid and its numbers from image

processed by Computer Vision (CV) library. If during the image processing, user presses "Back" button or "Main window" button, then the user will return to main window of Mobile Application. If the detection is completed successfully, then the detected numbers will be passed to Sudoku Solver activity and the current "Detect Sudoku" activity ends.

6.2 Graphical Design

6.2.1 UI/UX

For the User Interface (UI) colours, the majority of colour choices are based on orange colour and all of the buttons are given white text colour.
Logo of the Mobile Application is created at Logomakr webpage [1] and it is free to create and use custom logo [2].
In order to improve user experience, simple navigation is implemented, and it gives the user possibility to return to the main application window from any activity despite the main application window activity. Also, application indicates the active cell inside of the grid using blink animation.

[1] https://logomakr.com/, (accessed: 02.06.2020)
[2] https://my.logomakr.com/terms-of-use/, (accessed:07.06.2020)

6.2.2 Wireframe

The sketch of application layout is illustrated in Figure 6.3. There are 3 main screens: Main, Solver and Scan.
The starting point of the mobile application is the main screen and the further activities to be loaded depends on the button pressed by user. If "Load picture" button is pressed, then device gallery is opened, and user can choose the image to load for scanning. In case, "Take picture" button is pressed then device camera is launched, and user can take picture for scanning. Another option is to load .txt file, which is saved previously solved Sudoku and if .txt file is loaded than the sudoku grid is filled from the supplied file. Last option is to enter manually, where sudoku grid is loaded and user can enter the values manually.

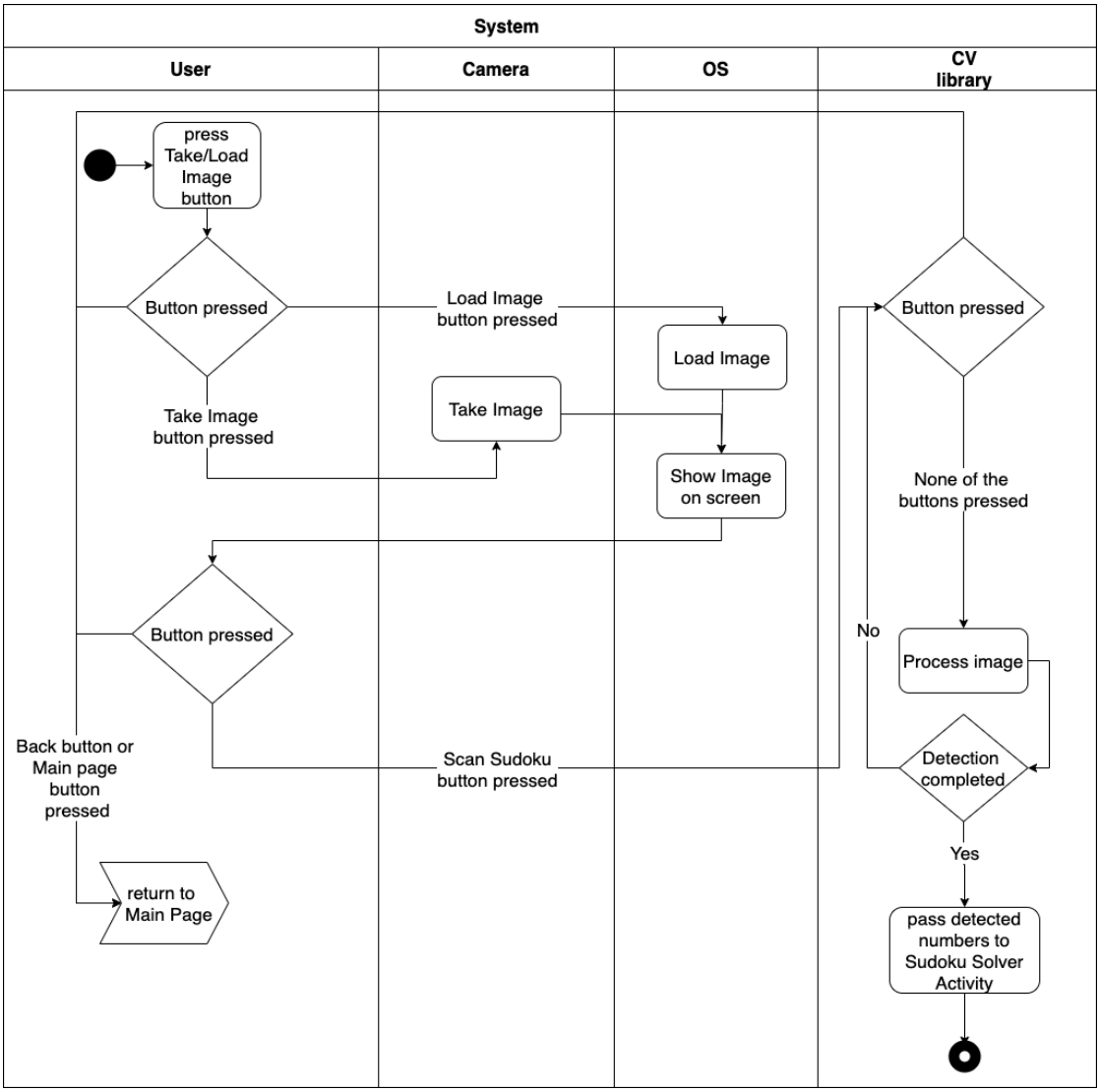


Figure 6.2: Activity diagram for use case "Detect Sudoku"

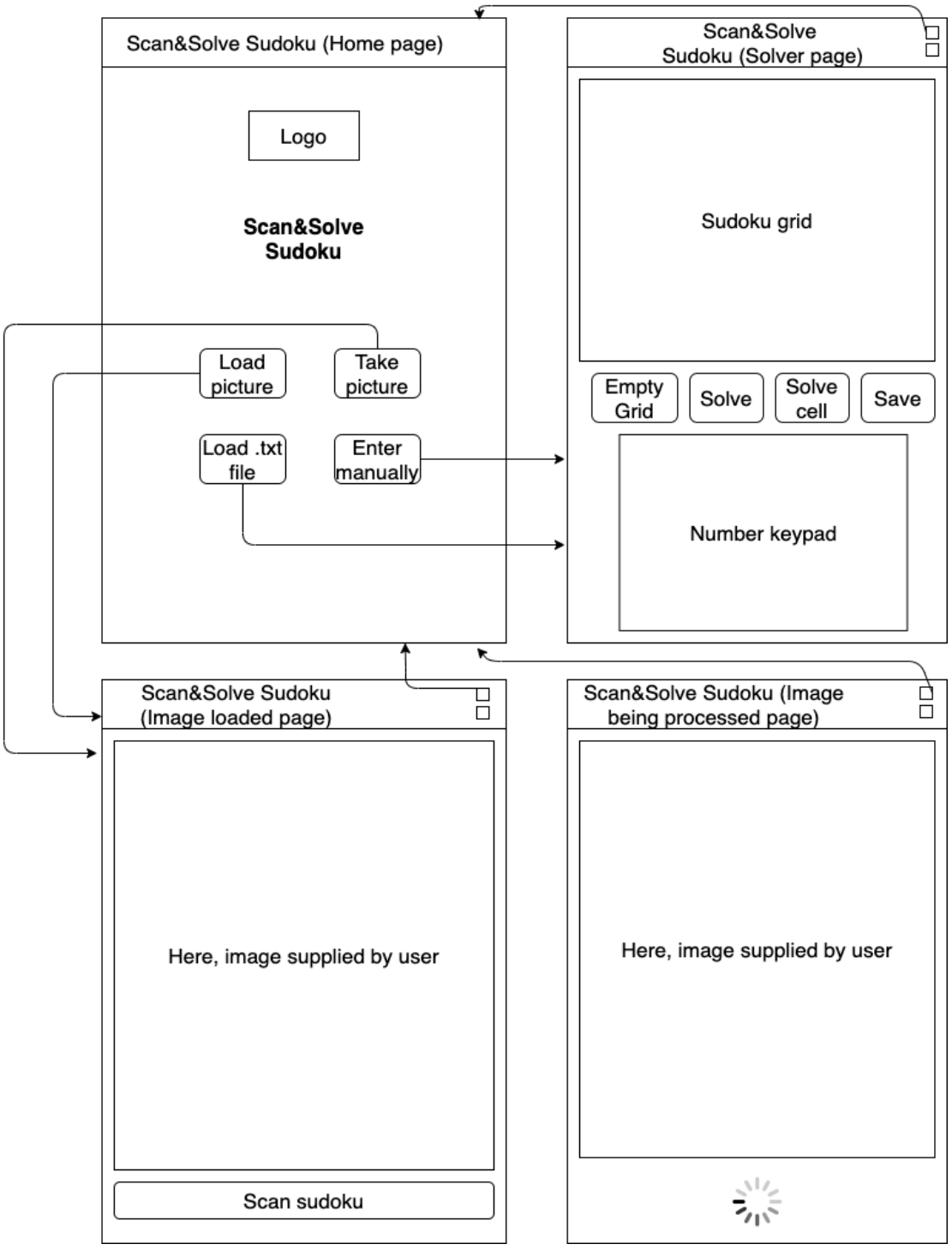


Figure 6.3: Wireframe of the application

7 Implementation

7.1 Setup and Configuration

Application development is accomplished using Android Studio IDE. For the purpose of using OpenCV, the stable version 3.4.3 of OpenCV Android Library was downloaded and imported into the project. Camera, read and write file permissions are requested in the Android manifest file.

7.2 Coding

Sudoku Scanning algorithm
The inspiration for the following Sudoku scanning implementation was partly taken from [1].

Preprocessing image
In general words, the supplied image is converted to gray scale as shown below:

```
1 Imgproc.cvtColor(colorimg, sudoku, Imgproc.COLOR_RGB2GRAY);
```

Listing 7.1: Color convert to grayscale

Then using Imgproc.GaussianBlur function the image is blurred a little in order to reduce the image noise:

```
1 Imgproc.GaussianBlur(sudoku, sudoku, new Size(11, 11), 0);
```

Listing 7.2: Image blurring

Since, image can have different light levels, adaptive threshold is applied and it will calculate the mean over 5x5 window and subtracts 2 from the mean, then the calculated threshold level is applied for each pixel in image.

[1] https://aishack.in/tutorials/sudoku-grabber-opencv-plot/, (accessed: 08.06.2020)

```
Imgproc.adaptiveThreshold(sudoku, outerBox, 255,
Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY, 5, 2);
```

Listing 7.3: Image thresholding

In order to get the Sudoku borders, the image is inverted, so that the borders of Sudoku and present noise are white.

```
Core.bitwise_not(outerBox, outerBox);
```

Listing 7.4: Image invertion

Since, thresholding can remove the connected parts as lines, dilating is applied to fill up those small missing image elements.

```
byte [] data = {0,1,0,1,1,1,0,1,0};
Mat ukernel = new Mat(3,3,CvType.CV_8U);
ukernel.put(0,0,data);
Imgproc.dilate(outerBox, outerBox, ukernel);
```

Listing 7.5: Image dilation

The result is shown in Figure 7.1.

Extracting Sudoku puzzle

Firstly, Sudoku bounding box is found by finding contours in image and the biggest image contour present in image is taken as Sudoku board. Afterwards, it is checked if extracted bounding box contains exactly four corners. If it contains, then the Sudoku board image is extracted after several steps of maths in order to calculate finer image dimensions to extract and image cosmetics are applied such as warping, which is used for correcting image distortions and cleaning lines using Hough lines, which basically is used to detect straight lines and after detection the lines are redrawn on image.
Figure 7.2 indicates the result after extracting Sudoku puzzle.

```
public Mat getSudokuArea(Mat image) {
       Mat preprocessed = preprocess(image);
       MatOfPoint poly = findBiggerPolygon(preprocessed);
       MatOfPoint2f aproxPoly = aproxPolygon(poly);

       if (Objects.equals(aproxPoly.size(), FOUR_CORNERS)) {
```

```
            int size = distance(aproxPoly);
            Mat cutted = applyMask(image, poly);
            Mat wrapped = wrapPerspective(size,
            orderPoints(aproxPoly), cutted);
            Mat preprocessed2 = preprocess2(wrapped);
            Mat withOutLines = cleanLines(preprocessed2);
            return withOutLines;
        }

        return preprocessed;
    }
```

Listing 7.6: Extracting Sudoku puzzle

Extracting Sudoku cells and detecting the numbers

First of all, the cell sizes are obtained by dividing image height by 9, afterwards for loop goes through rows, columns both in range of 0-8 (inclusive) and extracts rectangular pieces of cell size from image. All above-mentioned generalized steps are summarized from code in sample depicted in Listing 7.7. Afterwards, the K-NN (K-Nearest Neighbor) classification model identifies which number is the most suitable for the supplied sample, which in our case is extracted Sudoku cell. Listing 7.8 shows the usage of actual K-NN number identification method knn.findNearest(...) being used.
Figure 7.3 indicates the result after extracting Sudoku cell.

```
private List<Integer> extractCells(Mat m) {
        DetectDigit detect = null;
        try {
            Mat img = Utils.loadResource(this, R.drawable.digits,
            Imgcodecs.CV_LOAD_IMAGE_COLOR);
            detect = new DetectDigit(img);
        }catch (IOException e) {
            e.printStackTrace();
        }
        List<Mat> cells = getCells(m);
        List<Optional<Rect>> digitBoxes = Lists.transform(cells,
        FeatureDetector.GET_DIGIT_BOX_BYTE_SUM);

        List<Integer> result = Lists.newArrayList();
        List<Mat> cuts = Lists.newArrayList();

```

```
17          for(int i = 0; i < cells.size(); i++ ) {
18              Mat cell = cells.get(i);
19              com.google.common.base.Optional<Rect> box =
20              digitBoxes.get(i);
21
22              int d = 0;
23              if (box.isPresent() &&
24              CONTAIN_DIGIT_SUB_MATRIX_DENSITY.apply(cell)) {
25                  /* cut current cell to the finded box */
26                  Mat cutted = new Mat(cell, box.get()).clone();
27                  Imgproc.rectangle(cell, box.get().tl(),
28                  box.get().br(), Scalar.all(255));
29                  cuts.add(cutted);
30                  d = detect.detect(cutted);
31              }
32              Imgproc.rectangle(cell, new Point(0,0), new
33              Point(100,100), Scalar.all(255));
34              result.add(d);
35          }
36
37          Mat m2 = new Mat(0, cells.get(0).cols(), CvType.CV_8SC1);
38          for(Mat digit: cells) {
39              m2.push_back(digit.clone());
40          }
41          return result;
42      }
```

Listing 7.7: Extracting Sudoku puzzle cells

```
1 public Integer detect(Mat digit) {
2          Mat wraped = deskew(center(digit.clone()));
3          Mat result = new Mat();
4          Mat neighborhood = new Mat();
5          Mat distances = new Mat();
6
7          knn.findNearest(procSimple(wraped), 3, result, neighborhood,
8          distances);
9
10         System.out.printf("%s\n", neighborhood.dump());
11
12         return (int)result.get(0,0)[0];
```

```
13      }
```

Listing 7.8: Number identification method

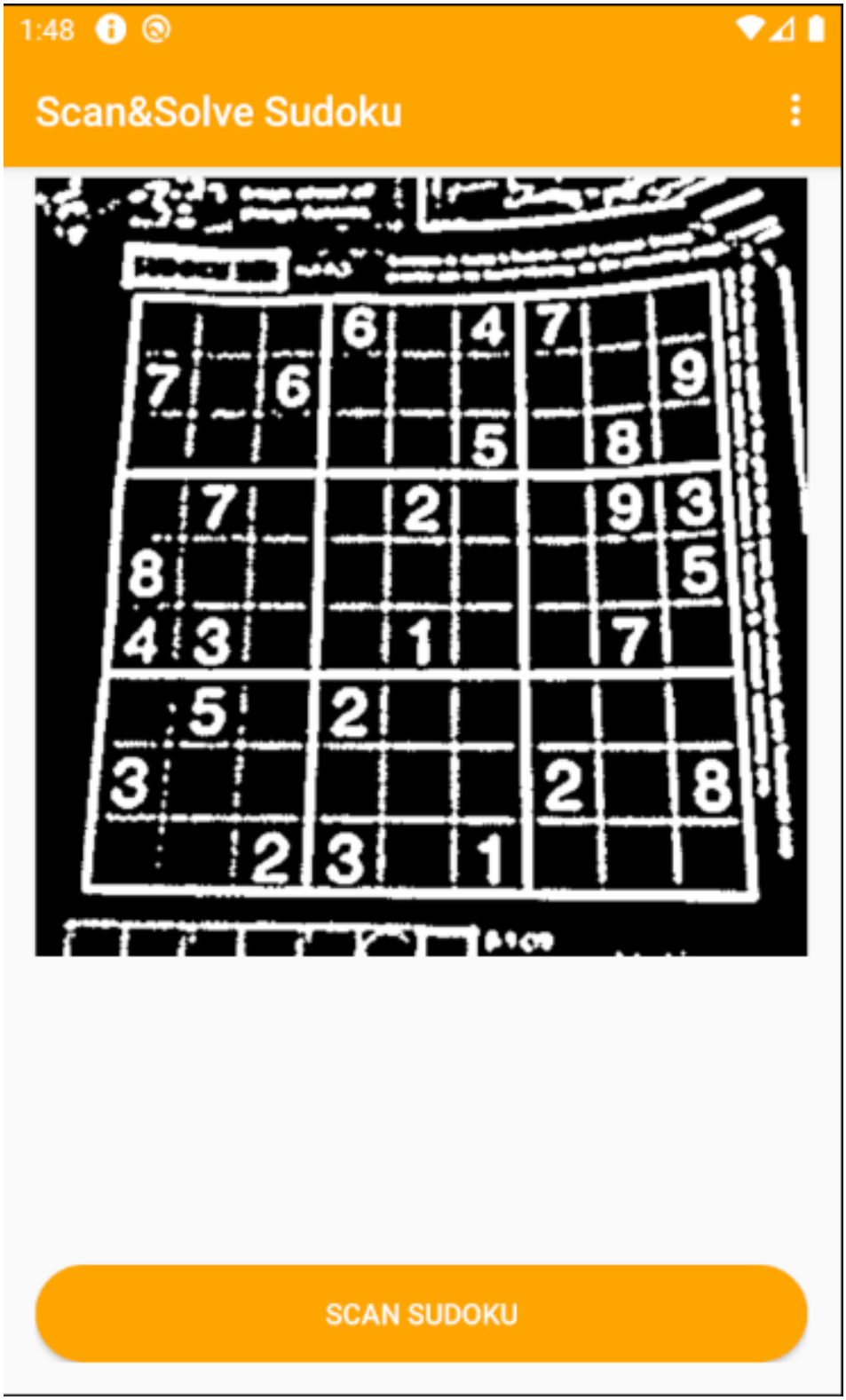


Figure 7.1: Screenshot after image preprocessing

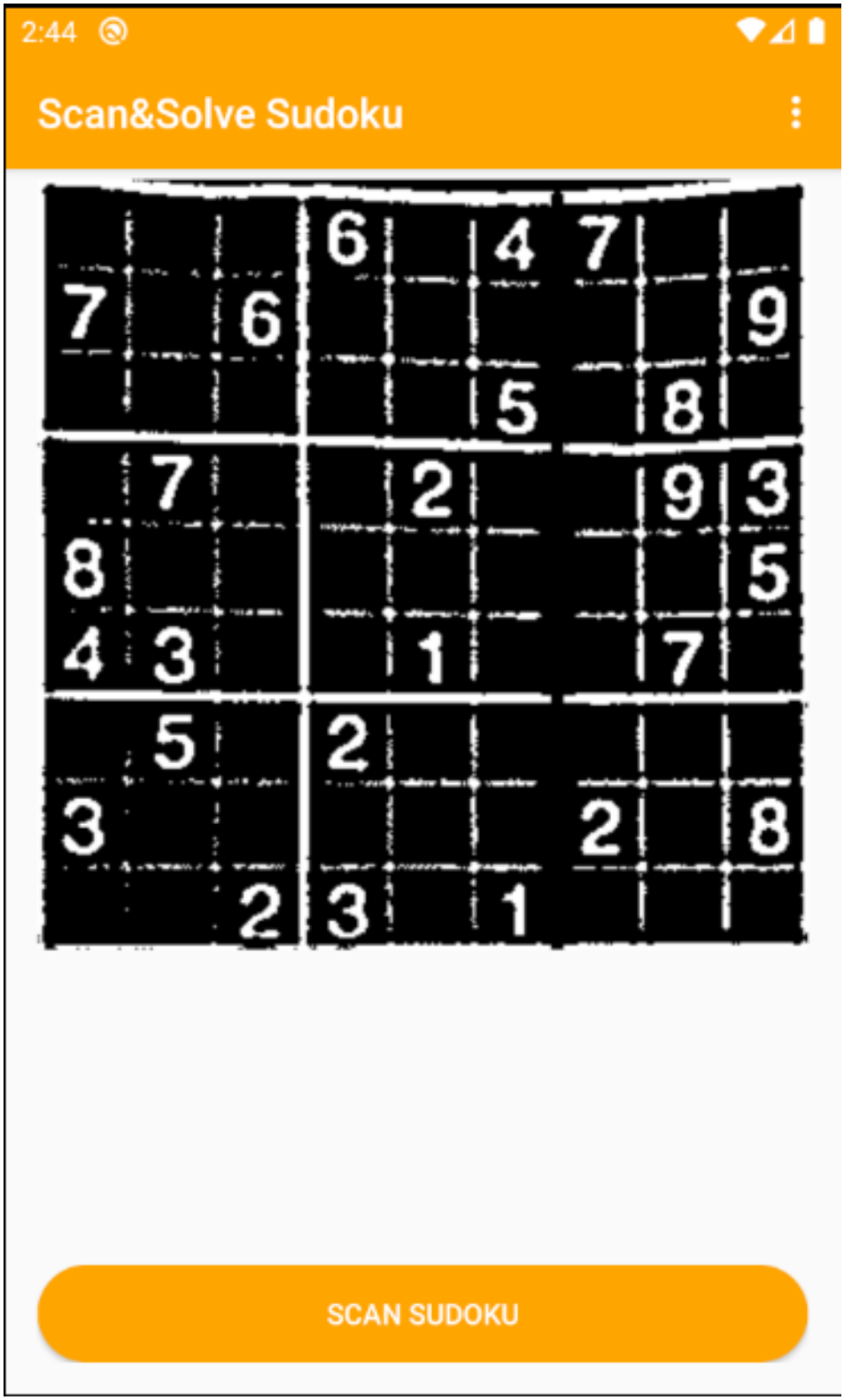


Figure 7.2: Screenshot after extracting Sudoku puzzle

Figure 7.3: Screenshot after extracting Sudoku cell

8 Evaluation

This chapter gives an overview on the topic of Software evaluation with respect to the number of factors that define the functionality and usability of software.

8.1 Unit tests

As it is obvious from the name of this section, the unit tests [11] examine the individual units (or in other words components) of the software that is the subject of testing. The main objective of conducting unit tests is the automated verification of software performance with a respect to the requirements of software product.

8.1.1 Definitions

By definition, a **unit** is the smallest testable part of software and in most of the cases, it has one or a few inputs and a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming [19], the methods are considered to be the smallest units. In general, the combination of the smallest testable units can build up the comprehensive tests for the complex applications.

8.1.2 Typical scenario

During the software testing stage, a software engineer implements the test cases, where one defines what shall be expected after certain number of operations inside the software program. During the test case execution, the developer will be notified about each test that fail or succeed through software run-time logs. Additionally, each log will be included in the final report. After the completion of the unit tests execution, the final report indicates the correctness of each tested unit of software with respect to the criteria, which was pre-defined inside of the test case.

8.1.3 Summary

As it has been mentioned, the unit tests bring the certain level of code quality and it is always appreciated to have such verification tool, because in a long run for the considerably long-lasting Projects the unit tests bring the huge impact as a software product scale expands. However, in order to guarantee the correct behaviour for every execution path, possible input and to ensure the absence of errors, the huge amount of time needs to be invested. For an instance, every Boolean decision statement requires at least two tests: one with an outcome of "true" and another with outcome "false". As a result, for every line of code written, software engineer has to implement 3 to 5 lines of test code. Undoubtedly, this requires considerable amount of time and its investment may not be worth the effort in a sense that if Project is a short term, then unit tests may be skipped or procrastinated for the later time point for a case, when there is a possibility for the Software product to be expanded for larger scale. For the current Android app, the unit tests are not implemented because it is out-of-scope and there was not sufficient amount of time to conduct the thorough unit testing of the app. Hence, the decision was to simply use the app and analyse the mistakes, which may emerge during the run time.

8.2 Evaluation with respect to the varying Android mobile devices

As it has been mentioned in the previous Chapters, there are numerous amounts of Android mobile devices available in the market and each device has different hardware and software parameters. In order to address this problem of varying Android devices, it has been decided to make the extensive use of Android emulated devices or in other words, the Android virtual devices, which are available inside the Android Studio development tool. This gives obvious advantage in the software evaluation due to the reason that it is not required to have in hand the physical hardware device for each desirable Android mobile device. On Android Studio, by changing the target device to any desired Android device, the whole app implementation may be instantiated in the chosen Android mobile device after the short waiting time, because normally Android Studio requires short amount of time for fetching the required device libraries.

8.3 Evaluation with respect to the pre-defined requirements in Chapter 4

8.3.1 Evaluation with respect to functional requirements

Considering the functional requirements defined in Chapter 4, all of the functional requirements have been addressed and one can find the functional requirements correlation with the implemented Android app, which has been deployed on Google PlayStore for open download. In the following paragraphs we will discuss in details the evaluation with respect to each functional requirement defined in Chapter 4, in particular in Table 4.10 and all of the below-mentioned functional requirements are referenced from Table 4.10.

F1: The software solution must solve Sudoku puzzle
Evaluation with respect to F1: The "ScanSolve Sudoku" Android application, which is live on PlayStore solves only the Sudoku puzzle and the whole implementation mainly concentrates on recognizing the numbers, solving Sudoku, importing Sudoku .txt file or manually entering the puzzle.

F2: The software solution must recognise the Sudoku board and numbers located inside of the board from picture
Evaluation with respect to F2: The implementation of the current Android Project proceeds as follows, first the Sudoku board is identified from picture and afterwards the numbers, which are located inside of the board are recognized.

F3: The user must be able to take picture
Evaluation with respect to F3: From the development side, in order to accomplish this task, the Camera permissions have been set to request every time if there is a new install of "ScanSolve Sudoku" app. From the user side, user has to give the permissions to use the camera and media file system, else app will not launch.

F4: The user must be able to import picture
Evaluation with respect to F4: From the development side, in order to accomplish this task, the media access permissions have been set to request every time if there is a new install of "ScanSolve Sudoku" app. From the user side, user has to give the permissions to use the camera and media file system, else app will not launch.

F5: The user must be able to load .txt file, which consists of the numbers only
Evaluation with respect to F5: From the development side, in order to accomplish this task, the file access permissions have been set to request every time if there is a new install of "ScanSolve Sudoku" app. From the user side, user has to give the permissions to use the camera and media file system, else app will not launch.

F6: The user must be able to input manually
Evaluation with respect to F6: In order to meet this requirement, the separate View and Activity have been implemented for inputting the Sudoku puzzle manually.

F7: The user must be able to save the solved question
Evaluation with respect to F7: From the development side, in order to accomplish this task, the file access permissions have been set to request every time if there is a new install of "ScanSolve Sudoku" app. From the user side, user has to give the permissions to use the camera and media file system, else app will not launch.

F8: The user must be able to sort the saved solution according to name, date
Evaluation with respect to F8: The file system of Android mobile device has the functionality to sort the .txt files according to different file parameters such as name, date, etc.

F9: The user must be able to filter the saved solution according to name, date
Evaluation with respect to F9: The file system of Android mobile device has the functionality to filter the .txt files according to different file parameters such as name, date, etc.

F10: The user must be able to get hint on next random cell
Evaluation with respect to F10: There is a button "Solve cell" implemented for this purpose, which gives solution for the next random cell.

F11: When user presses solve button, the software must check user input for the wrong values entered inside of grid cells and indicate the wrong grid cells
Evaluation with respect to F11: There is a button "Solve" implemented for this purpose and when this button is pressed, firstly it will check the correctness of the values entered inside of grid cells and when the incorrect values are found, they are

indicated by the red color.

F12: When clicked on "Solve" button, if the wrong cells are not found, the Software shall give the full correct solution
Evaluation with respect to F12: There is a button "Solve" implemented for this purpose and when this button is pressed and no incorrect values inside Sudoku board cells are found, then the solution for all of the Sudoku board cells is given.

F13: When clicked on "Empty" button, all of the grid cells shall be cleared
Evaluation with respect to F13: There is a button "Empty" implemented for this purpose and when this button is pressed, all of the values inside Sudoku board cells are cleared.

8.3.2 Evaluation with respect to non-functional requirements

Coming to the non-functional requirements, despite of NF2 all of the non-functional requirements have been met as well. The details of the evaluation with respect to each non-functional requirement defined in Chapter 4, in particular in Table 4.11 are discussed in below paragraphs.

NF1: The solution of Sudoku puzzle shall be ready maximum in 1.2 seconds, this maximum time is given, because the Software performance differs slightly on different hardware.
Evaluation with respect to NF1: The "ScanSolve Sudoku" Android application tests on 10 Android devices showed that the solution of Sudoku puzzle have always been lower than 1.2 seconds and on average it was 1.0 seconds.

NF2: The user acceptance test shall be defined for the speed of the identification of objects on picture, because each mobile hardware device has varying performance parameters.
Evaluation with respect to NF2: This non-functional requirement has not been met, because of the strict time constraints.

NF3: This software solution, which is based on Android platform must support the

minimum sdk version for Image Processing library and above API levels.
Evaluation with respect to NF3: The minimum sdk version of "ScanSolve Sudoku" Android application is 16, which supports the minimum sdk version for OpenCV Image Processing library. The compile sdk version is 28, which corresponds to Pie code name of Android platform. It means, this mobile application can be installed on all Android devices with API level from 16 up to including 28.

NF4: If software solution oriented on mobile devices, no external device shall be required.
Evaluation with respect to NF4: The current Android mobile application does not require any external device due to the reason that all of the necessary mobile hardware is already built-in inside the vast majority of mobile devices.

NF5: Programming language shall be Java.
Evaluation with respect to NF5: The implementation of the current Android mobile application is based on Java programming language and it can be easily verified by checking the source code.

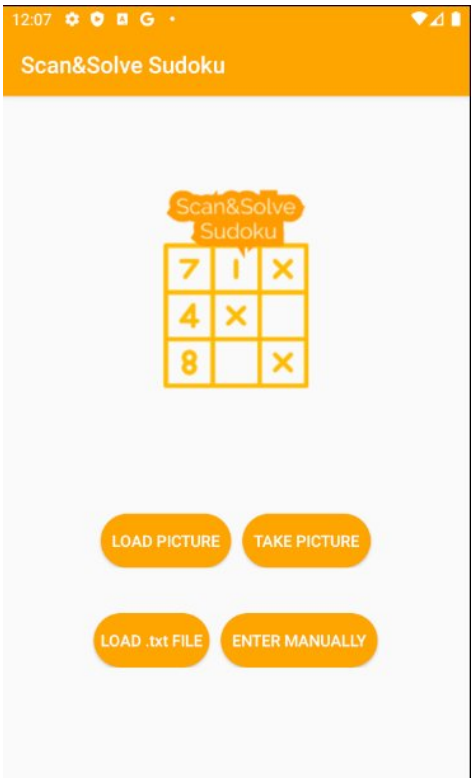
9 Summary

This chapter summarizes the results obtained, problems faced and how they are tackled. As it comes to results, it should be noted that the completed Android mobile application is deployed at Google PlayStore, which is accessible at [1] and the source code for the latest app releases will be uploaded to my public Github repository [2]. The future work section concerns the new app releases and the conclusion section gives the brief overview of this Thesis work.

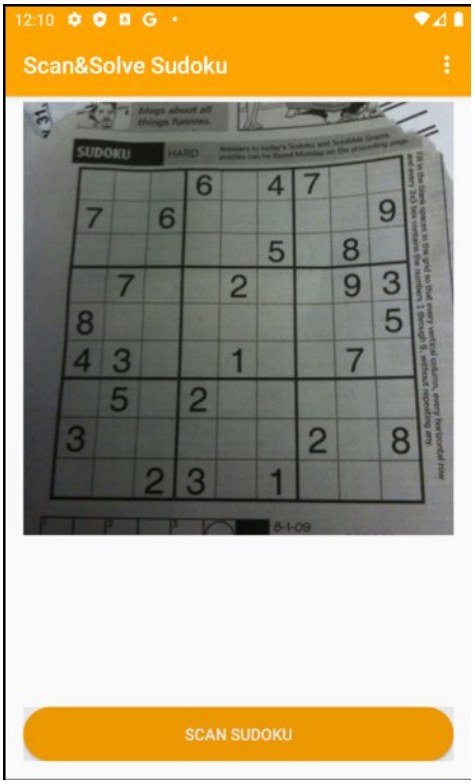
9.1 Results

Current state of the mobile application allows the user to trigger application using file, image upload, manual input or by taking photo with a device camera. The end result is either Sudoku board and its numbers being identified and being forwarded to Sudoku solver screen or manual input and automated solution of Sudoku puzzle. Despite the fact that considerable amount of work has been done on all the current Project related fields, not all functional requirements were fully met. The main encountered issues were limited amount of time and lack of experience in the main development tools and libraries such as Android, OpenCV and Machine Learning techniques. Also, another big portion of the project time was spent on its documentation, which in a matter of fact is the current Thesis paper. Owing to the reason that the end product is a prototype, accuracy tests were not conducted. However, the product will be updated with more features, more accurate results and better User Experience (UX). The screenshots of the "ScanSolve Sudoku" Android mobile application are shown on pages from 62 to 65 (inclusive).

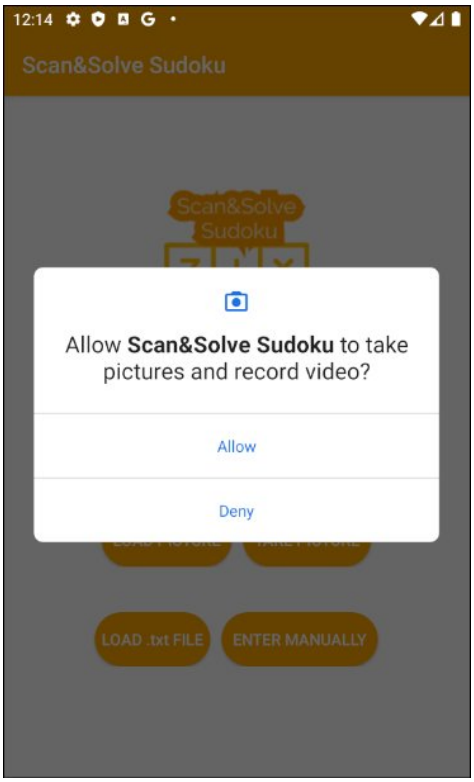
[1] https://play.google.com/store/apps/details?id=com.shukhratKhaydarov.sudoku, (accessed: 17.07.2020)
[2] https://github.com/SeanKh/ScanSolveSudoku (accessed: 01.07.2020)



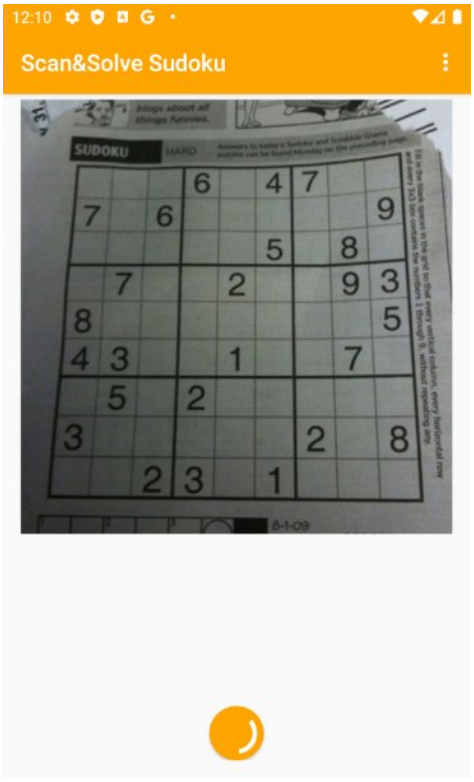
(a) Screenshot of home page



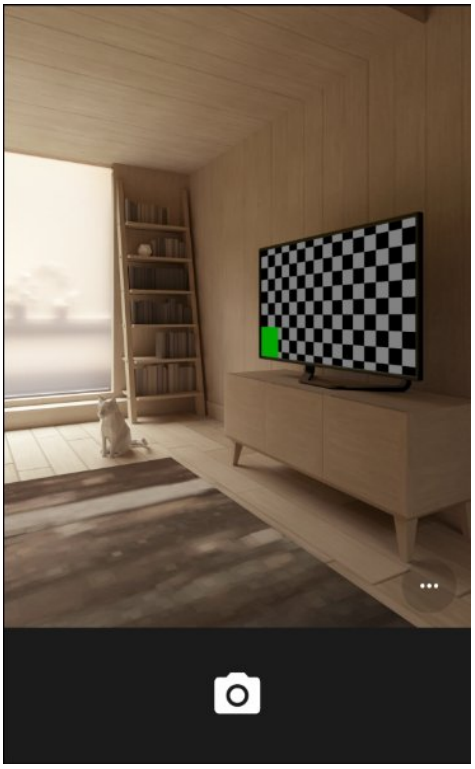
(b) Screenshot of loaded Sudoku image



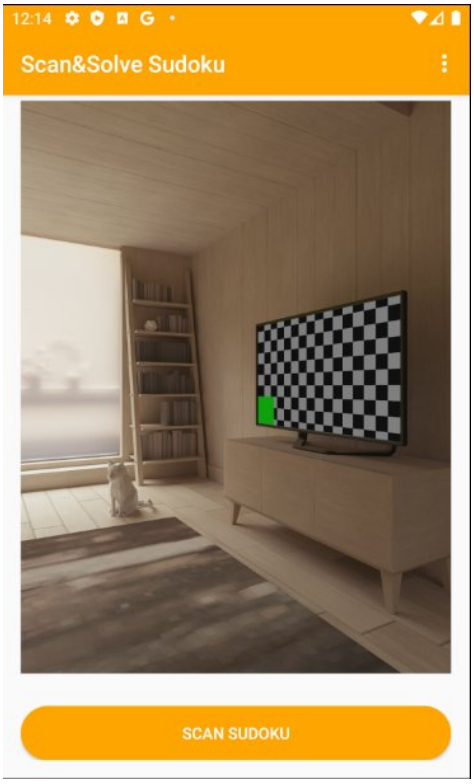
(c) Screenshot of camera permissions being requested



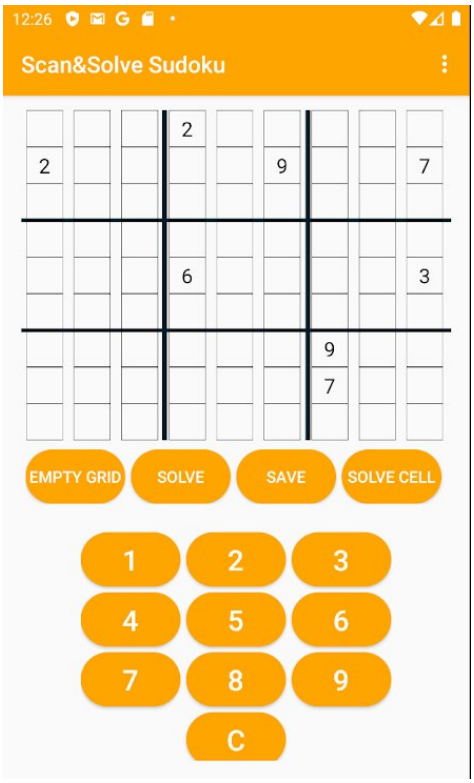
(d) Screenshot of Sudoku image being processed



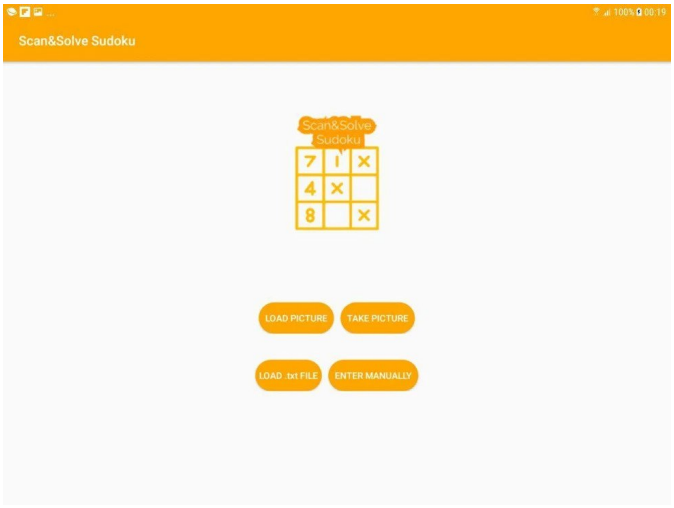
(a) Screenshot of the image being shot using camera



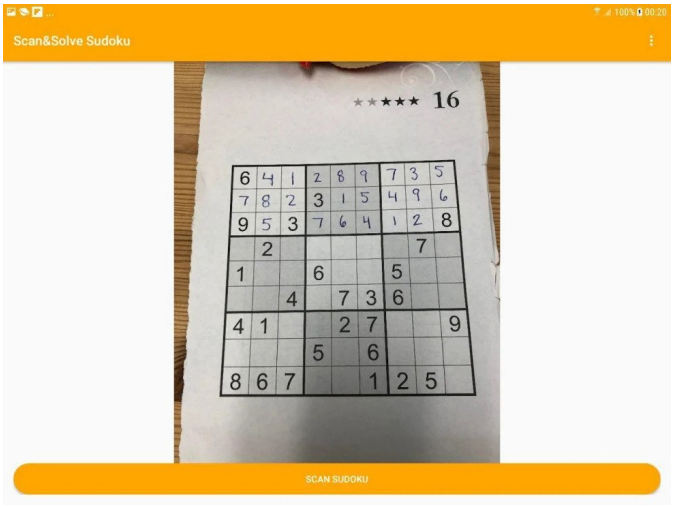
(b) Screenshot of the picture after taking the picture using camera



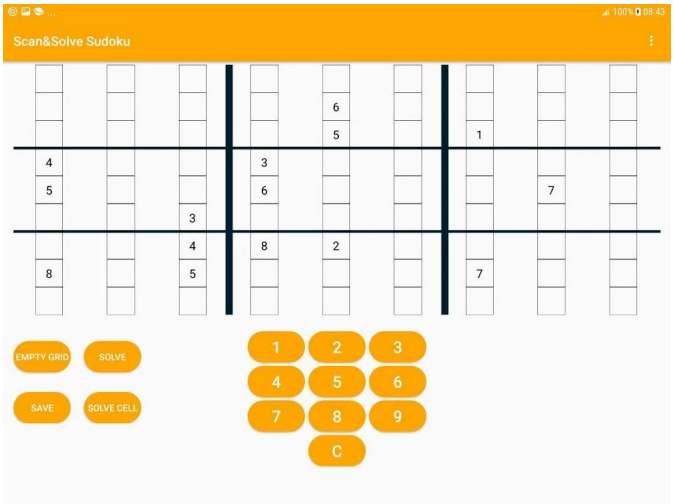
(c) Screenshot of Sudoku board with numbers entered



(a) Screenshot of home page on tablet screen



(b) Screenshot of Sudoku image being loaded on tablet screen



(c) Screenshot of Sudoku board with numbers entered on tablet screen

9.2 Future work

Future work of the current product is concerned with implementing more sophisticated logic and Image Processing techniques in order to extract and deliver the images of number located inside Sudoku grid cell that shall meet the Classification model expectations for the image to be identified. Another point of improvement is to use more lightweight and effective ML Classification model in order to speed up the number guessing process.

9.3 Conclusion

Taking into account that final product is a prototype, the conclusions of this paper indicate that the problem of detecting and solving logic puzzles such as Sudoku can be tackled using mobile device capabilities.

Bibliography

[1] Baggio, Daniel L.: *OpenCV 3.0 computer vision with Java.* Packt Publishing Ltd, 2015

[2] Berglund, Tim ; McCullough, Matthew: *Building and testing with Gradle.* O'Reilly Media, Inc., 2011

[3] Burger, Wilhelm ; Burge, Mark J.: Binary images. In: Fagerberg, Jan (Hrsg.) ; Mowery, David C. (Hrsg.) ; Nelson, Richard R. (Hrsg.): *Digital image processing: an algorithmic introduction using Java.* Oxford : Springer, 2016, Kap. 1, S. 11

[4] Burger, Wilhelm ; Burge, Mark J.: Color images. In: Fagerberg, Jan (Hrsg.) ; Mowery, David C. (Hrsg.) ; Nelson, Richard R. (Hrsg.): *Digital image processing: an algorithmic introduction using Java.* Oxford : Springer, 2016, Kap. 1, S. 11

[5] Burger, Wilhelm ; Burge, Mark J.: Color Spaces and Color Conversion. In: Fagerberg, Jan (Hrsg.) ; Mowery, David C. (Hrsg.) ; Nelson, Richard R. (Hrsg.): *Digital image processing: an algorithmic introduction using Java.* Oxford : Springer, 2016, Kap. 12, S. 303

[6] Burger, Wilhelm ; Burge, Mark J.: Grayscale images (intensity images). In: Fagerberg, Jan (Hrsg.) ; Mowery, David C. (Hrsg.) ; Nelson, Richard R. (Hrsg.): *Digital image processing: an algorithmic introduction using Java.* Oxford : Springer, 2016, Kap. 1, S. 10–11

[7] Burger, Wilhelm ; Burge, Mark J.: HSV/HSB and HLS Color Spaces. In: Fagerberg, Jan (Hrsg.) ; Mowery, David C. (Hrsg.) ; Nelson, Richard R. (Hrsg.): *Digital image processing: an algorithmic introduction using Java.* Oxford : Springer, 2016, Kap. 12, S. 306

[8] Craig, Clifton ; Gerber, Adam: *Learn Android Studio: Build Android Apps Quickly and Effectively.* Apress, 2015

[9] Derpanis, Konstantinos G.: *Overview of binomial filters.* 2005

[10] Gonzalez, R.C.: *Digital Image Processing 2Nd Ed.* Prentice-Hall Of India Pvt. Limited, 2002. – URL https://books.google.de/books?id=iyJOPgAACAAJ. – ISBN 9788120327580

[11] Hamill, Paul: *Unit test frameworks: tools for high-quality software development.* " O'Reilly Media, Inc.", 2004

[12] Holzer, Adrian ; Ondrus, Jan: Mobile app development: Native or Web? In: *Proc. Workshop eBus.(WeB)*, 2012

[13] Huo, Ming ; Verner, June ; Zhu, Liming ; Babar, Muhammad A.: Software quality and agile methods. In: *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.* IEEE (Veranst.), 2004, S. 520–525

[14] Kamal, Snigdha ; Chawla, Simarpreet S. ; Goel, Nidhi: Detection of Sudoku puzzle using image processing and solving by Backtracking, Simulated Annealing and Genetic Algorithms: A comparative analysis. In: *2015 third international conference on image information processing (ICIIP)* IEEE (Veranst.), 2015, S. 179–184

[15] Kotsiantis, Sotiris B. ; Zaharakis, I ; Pintelas, P: Supervised machine learning: A review of classification techniques. In: *Emerging artificial intelligence applications in computer engineering* 160 (2007), Nr. 1, S. 3–24

[16] Saraosos, Jonas Reycian B. ; Regis, Michael Anthony Jay B.: Development of an Android-Based Visual Sudoku Solver Using Contour Finding and Backtracking Algorithm. In: *Journal of Science, Engineering and Technology* 6 (2018), S. 208–217

[17] Sheikh, Aijaz A. ; Ganai, Prince T. ; Malik, Nisar A. ; Dar, Khursheed A.: Smartphone: Android Vs IOS. In: *The SIJ Transactions on Computer Science Engineering & its Applications (CSEA)* 1 (2013), Nr. 4, S. 141–148

[18] Simha, P. J. ; Suraj, K. V. ; Ahobala, T.: Recognition of numbers and position using image processing techniques for solving Sudoku Puzzles. In: *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, 2012, S. 1–5

[19] Xie, Tao ; Taneja, Kunal ; Kale, Shreyas ; Marinov, Darko: Towards a framework for differential unit testing of object-oriented programs. In: *Second International Workshop on Automation of Software Test (AST'07)* IEEE (Veranst.), 2007, S. 5–5

A Appendix

The CD, which is attached to the current Thesis paper contains the Thesis paper in .pdf-Format, Latex and Android Studio projects. The current Thesis paper may be viewed upon application to the principal examiner.

Declaration

I declare that this Bachelor Thesis has been completed by myself independently without outside help and only the defined sources and study aids were used.

City	Date	Signature