Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Ларикової Марії Володимирівни*
(ПІБ)

академічної групи *122-18-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка мобільного застосунку для автоматичного розв'язування судоку на основі Python*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О.В.*			
розділів:				
спеціальний	*доц. Реута О.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* *Ларикова М.В.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка мобільного застосунку для автоматичного розв'язування судоку на основі Python*

затверджена наказом ректора НТУ «ДП» від *18 травня 2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Реута О.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Ларикова М.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 88 с., 39 рис., 12 табл., 4 дод., 15 джерел.

Об'єкт розробки: мобільний застосунок для вирішення «Судоку» за допомогою камери.

Мета кваліфікаційної роботи: розробка застосунку для мобільних пристроїв з метою вирішення за допомогою комп’ютерного зору гри «Судоку» з камери або файлу зображення (формати файлів .jpg, .jpeg, .png й тощо), скорочення часу на пошук рішення гри, перевірки здатності вирішити пазл.

У вступі розглянуто актуальність проблеми, її стан та аналіз рішень, проаналізовано галузь застосування та необхідність вирішення проблеми, конкретизовано завдання кваліфікаційної роботи .

У першому розділі визначено та досліджено предметну галузь об’єкту розробки, знайдено підстави для початку розробки та виявлено необхідні, сучасні, доцільні технології та програмні засоби для програмної реалізації.

У другому розділі було описано технічне та програмне забезпечення, яке необхідно для розробки та роботи застосунку. Також було проаналізовано обрані технології та зроблені висновки щодо планування розробки інтерфейсу користувача.

У третьому розділі було визначено тривалість розробки, кількість спеціалістів, їх заробітна плата та бюджет на розробку застосунку.

Практичне значення полягає у розробці застосунку задля можливості вирішення головоломки «на льоту», маючи для цього лише зображення матриці 9 на 9 в будь-якому вигляді – цифровому або паперовому.

Актуальність цього додатку аналізується у декількох розділах та приводить до висновку про відсутність великої кількості аналогів та натякає про сучасність обраних технологій.

Список ключових слів: застосунок, судоку, Python, OpenCV, камера, комп’ютерний зір.

ABSTRACT

Explanatory note: 88 pp., 39 fig., 12 tables, 4 additions, 15 sources.

Object of development: a mobile application for solving "Sudoku" with the help of a camera.

Purpose of the qualification work: development of an application for mobile devices to solve with the help of computer vision of the game "Sudoku" from the camera or image file (file formats .jpg, .jpeg, .png, etc.), reducing the time to find a solution, verification ability to solve puzzles.

In the introduction the urgency of the problem, its state and analysis of solutions are considered, the field of application and the need to solve the problem are analyzed, the tasks of qualification work are specified.

In the first section, the subject area of the object of development is identified and researched, the grounds for the beginning of development are found and the necessary, modern, expedient technologies and software tools for software implementation are identified.

The second section described the hardware and software required to develop and operate the application. Selected technologies were also analyzed, and conclusions were drawn regarding the planning of user interface development.

The third section determined the duration of development, the number of specialists, their salaries, and the budget for application development.

The practical value is to develop an application to solve the puzzle "on the fly", having only the image of the matrix 9 by 9 in any form – digital or paper.

The relevance of this application is analyzed in several sections and leads to the conclusion that there are not many analogues and hints at the modernity of the selected technologies.

Keyword list: application, sudoku, Python, OpenCV, camera, computer vision.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – персональний комп’ютер;

VCS – Version Control System;

IT – Information Technologies;

UX – User Experience;

UI – User Interface;

Brute-force – метод грубої сили, повний перебір;

DLX – Dancing Links;

MVC – Model-View-Controller;

IDE – Integrated Development Environment;

CAC – Create Acceptance Criteria;

AAC – Analyze Acceptance Criteria;

ACR – Acceptance Criteria Review;

TechD – Technical Design;

TestD – Test Design;

BEi – Backend Investigation;

BE – Backend (Functionality Development);

BER – Backend Review;

UIi – Frontend Investigation;

UI – Frontend (Interface Development);

UIR – Frontend Review;

ATQCi – Test Investigation;

QC – Manual Testing;

AT – Automation Testing.

ЗМІСТ

ВСТУП

Судоку – це дуже популярна японська гра, яка виникла ще у 18 столітті та набула своєї популярності лише у 21 столітті завдяки тому, що це суміш літерного кросворду та чисел [1]. Для японців було дуже складно створювати кросворди через складову абетки, тому ідеальним варіантом стала головоломка, яка використовує числа замість літер.

Існує багато варіантів гри судоку, але класичним прийнято звати варіант судоку, де потрібно заповнювати порожні клітини цифрами від одного до дев'яти таким чином, щоб в жодному блоці три на три або рядку чи стовпчику не було повторень.

Ідея розробленого мобільного застосунку полягає у виростанні його у сфері розваг, ігор та головоломок. Існує багато вирішувачів для різних видів ігор і судоку не є виключенням. Такі вирішувачі використовуються не тільки щоб скоріше вирішити головоломку, але й для того щоб перевірити валідність вигаданого завдання або рішення.

З новими технологіями такі додатки також розвиваються, переходячи до роботи з комп'ютерним зором. Кожна поважаюча себе машина повинна вміти вирішувати різні головоломки, а кожна сучасна машина може вирішувати її не тільки за допомогою введення даних, але навіть з камери.

Тому дана кваліфікаційна робота є актуальною та має дуже велике значення для відображення процесу сучасного розвитку технологій. Метою такої роботи є не лише розробка мобільного додатку для вирішення гри «Судоку», а ще й вивчення нових технологій.

Щоб досягти поставленої мети необхідно вирішити основні проблеми:

- Проаналізувати аналогічні рішення;
- На основі аналізу існуючих рішень визначитися з вимогами до додатку та дизайном;
- Проаналізувати вимоги до додатку та обрати технології для швидкої та якісної реалізації;

– Спираючись на обрані технології спроектувати архітектуру додатку (технічний дизайн);
– Закласти певний час та створити план дій на розробку окремих частин проекту (окремих функцій додатку);
– Визначити та проаналізувати ризики, особливо, ті які пов’язані з необізнаністю з технологіями;
– Скорегувати у разі необхідності план дій для досягнення головної мети – створення додатку для вирішення головоломки.

Нижче наведені основні функційні задачі, які поставленні для цього проекту:

– Введення та виведення даних на екран;
– Вирішення головоломки з введених даних у застосунок;
– Вирішення головоломки з зображення переданого у застосунок;
– Вирішення головоломки за допомогою комп’ютерного зору;
– Виведення застережень при невірному введені даних.

Застосунок повинен працювати у мобільному застосунку і на комп’ютері. Якість зору додатку може вагатися в залежності від обраного набору даних для аналізу зображень.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Проаналізувавши за допомогою веб-застосунку від Google, «Google Trends» [2], запит «Sudoku» (Рис 1.1) було виявлено, що наразі існує деякий ріст тренд на цей запит за останні 12 місяців.

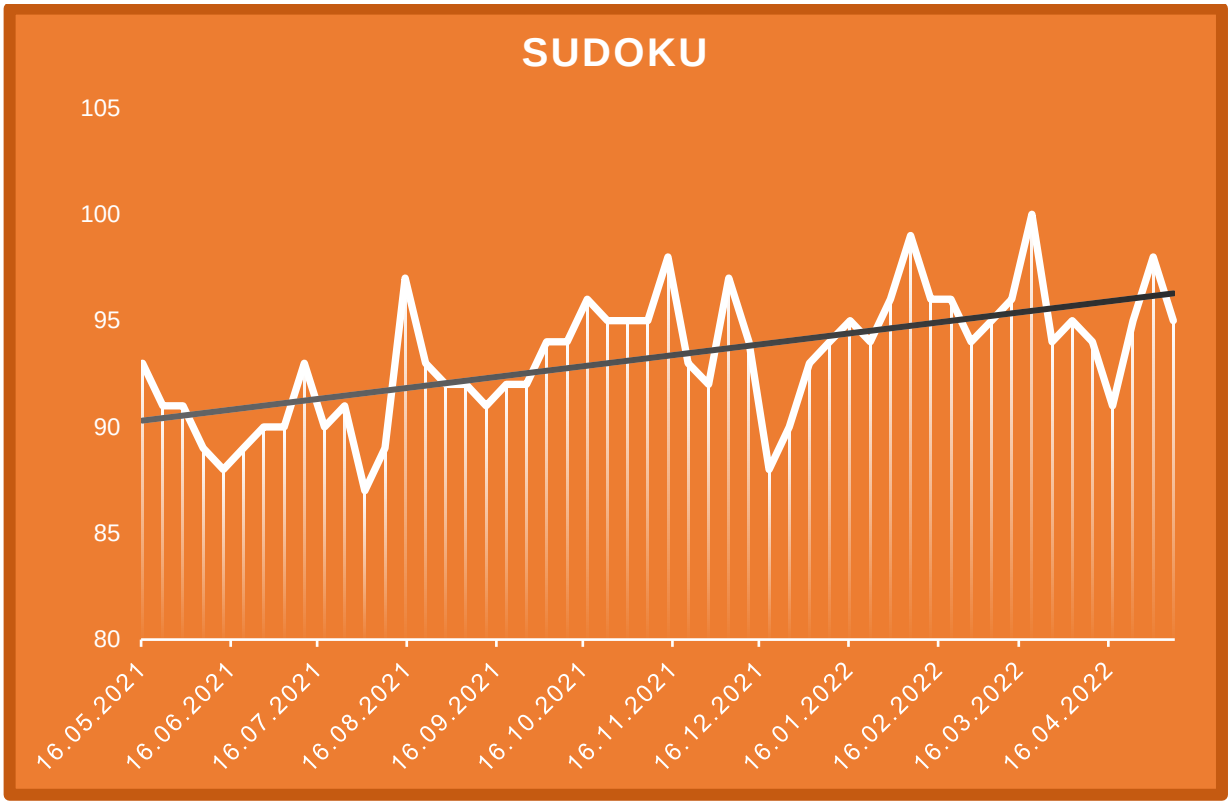

Рис 1.1. Результати аналізу запиту «Sudoku» у Google Trends

З цього робимо висновок про певну популярність на цю гру та її рішення. Тому можна перейти до аналізу аналогічних рішень у мережі інтернет.

У результаті пошуку таких самих додатків серед вирішальників було знайдено:

– Дуже багато вирішальників онлайн з вводу користувача (Рис 1.3);
– Декілька варіантів рішення за допомогою камери або зображення;
– Лише одне рішення, яке поєднує усі три варіанти в одне (Рис 1.2).

Нижче наведені скріншоти прикладів, які було знайдено у мережі інтернет.

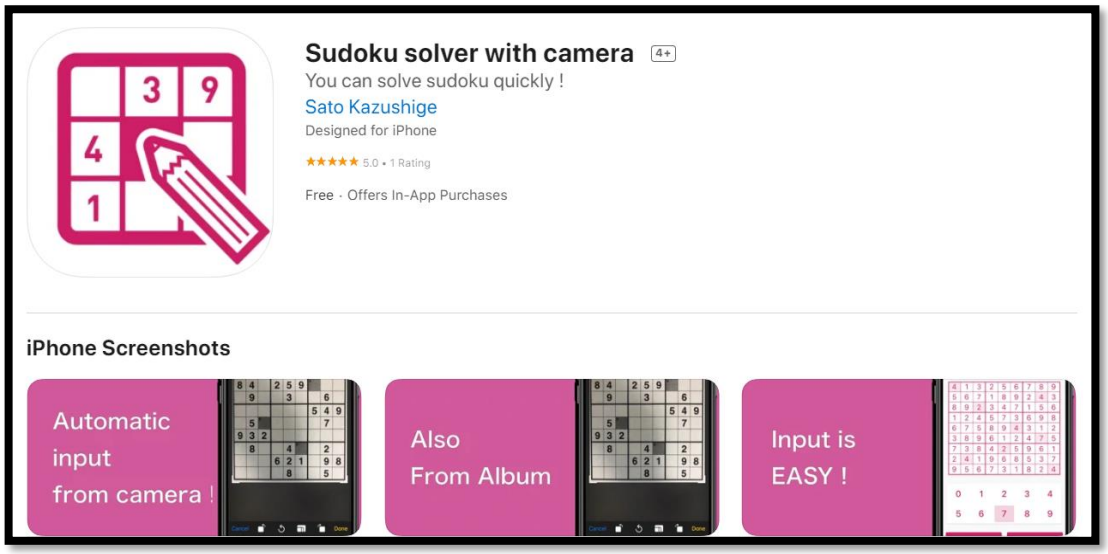

Рис 1.2. Приклад єдиного додатку повного аналогу планованого [3]

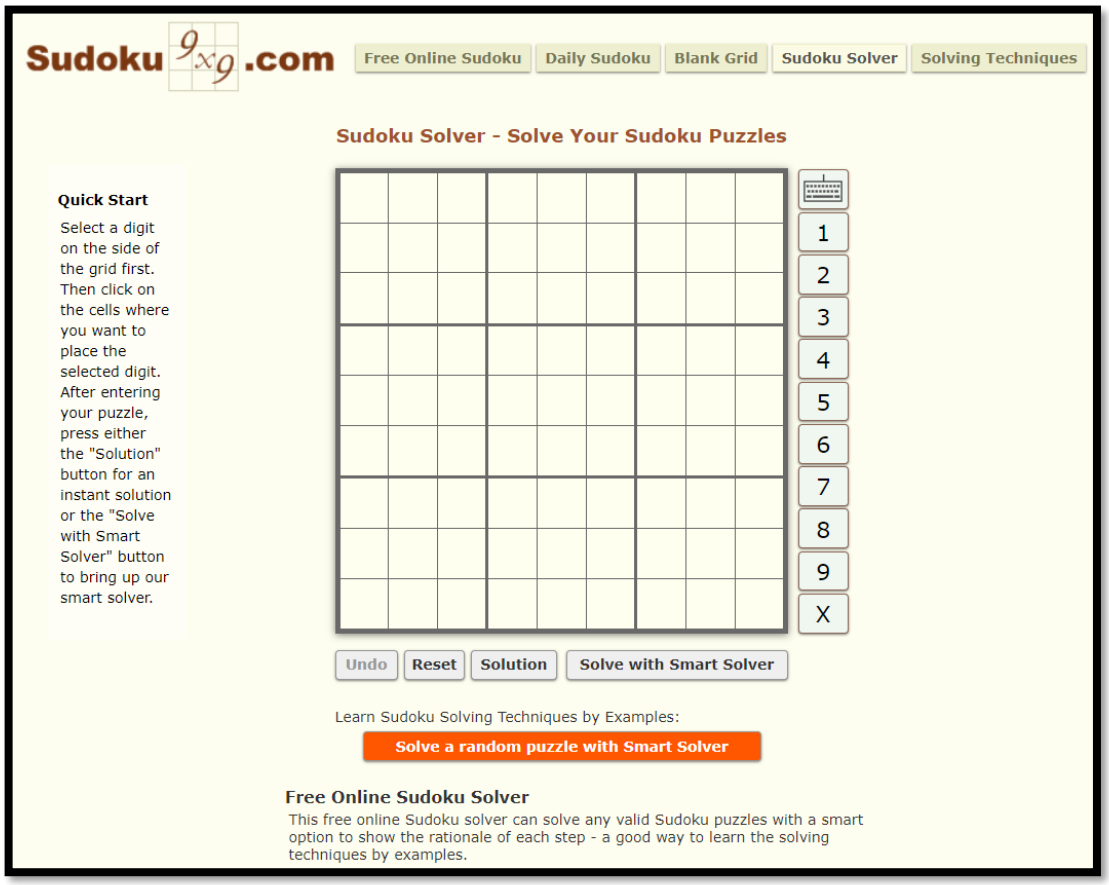

Рис 1.3. Приклад одного з багатьох веб-вирішальників [4]

Тобто це каже про недостатню ступінь рішення та відсутність великої кількості аналогів, що приводить до висновку про можливу необхідність створення свого рішення.

Наступний крок, визначити вимоги до додатку, знайти прогавини у технологіях та виявити технічні протиріччя з боку технічних рішень. Оскільки лише один додаток має усі три варіанти вирішення судоку, то його буде розглянуто як «еталон».

З вимог, які не були реалізовані у «еталоні» - це доступність додатку у декстопній версії, та для підтримки Android-систем. З наукової та технологічної точки зору не можна казати про які-небудь недоліки – на момент написання роботи застосунок не був протестований виконавцем кваліфікаційної роботи. Окрім цього, жодних відгуків на сайті не було, лише едина максимальна оцінка, що каже про проблеми додатку або його непопулярність через прогавини або його вигляд.

1.2 Призначення розробки та галузь застосування

Мобільний додаток для знаходження рішення пазлу гри «Судоку» застосовується у галузі розваг. Така галузь має дуже великий попит, але перетинути поріг входження до цієї галузі дуже важко через масштаби конкуренції та кількість вибору для користувача. Через це потрібно чітко розуміти попит на ринку та що можна протиставити аналогічним рішенням, щоб переманити потенційного клієнта.

1.3 Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

– освітня програма 122 «Комп’ютерні науки»;

- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18 травня 2022 р.;
- завдання на кваліфікаційну роботу на тему «Розробка мобільного застосунку для автоматичного розв’язування судоку на основі Python».

1.4 Постановка завдання

Завданням кваліфікаційної роботи є розробка мобільного додатку для вирішення головоломки «Судоку».

Кожен додаток має інтерфейс, до якого звертається користувач, він повинен бути зроблений по стандартам UX та UI.

Спершу було розроблено мокапи, які потребують дуже багато часу на реалізацію та є великий ризик не встигнути досягти головних функційних вимог. Тому було прийнято рішення спростити дизайн (Рис 1.4, Рис 1.5), щоб вкластися в строки виконання завдання для демонстрації повного функціоналу додатку. Після узгодження з клієнтом кінцевого результату, можна буде в подальшому зробити інтерфейс за першим варіантом дизайну.

Для застосунку обрати теплі кольори переважно помаранчевого кольору, який асоціюватиметеся з такими прикметниками як «грайливий», «привабливий», «веселий». Усі елементи інтерфейсу зробити розбірливими та розділеними. Головні кнопки розташувати по правий бік додатку. Кожна кнопка має підпис, який відповідає сенсу дії, яку виконує застосунок при натисканні тієї самої кнопки. Протилежним кольором до помаранчевого обрати чорний та його відтінки, таким чином написи на кнопках будуть розбірливими, а ключові елементи будуть привертати увагу до себе.

Зробити фон інтерфейсу відтінку чорного та протиставити написи на ньому у білих кольорах. Дошка для вводу підписана, а цифри на дошці відображені стандартним шрифтом, який є на пристрої з додатком.

Додати футер до додатку з цитатою про гру, яку сказав Шарью: «Confidence is when you stop using a PENCIL while solving SUDOKU.»[1] У перекладі вона буде звучати так: «Впевненість – це коли ти перестаєш використовувати ОЛІВЕЦЬ вирішуючи СУДОКУ».

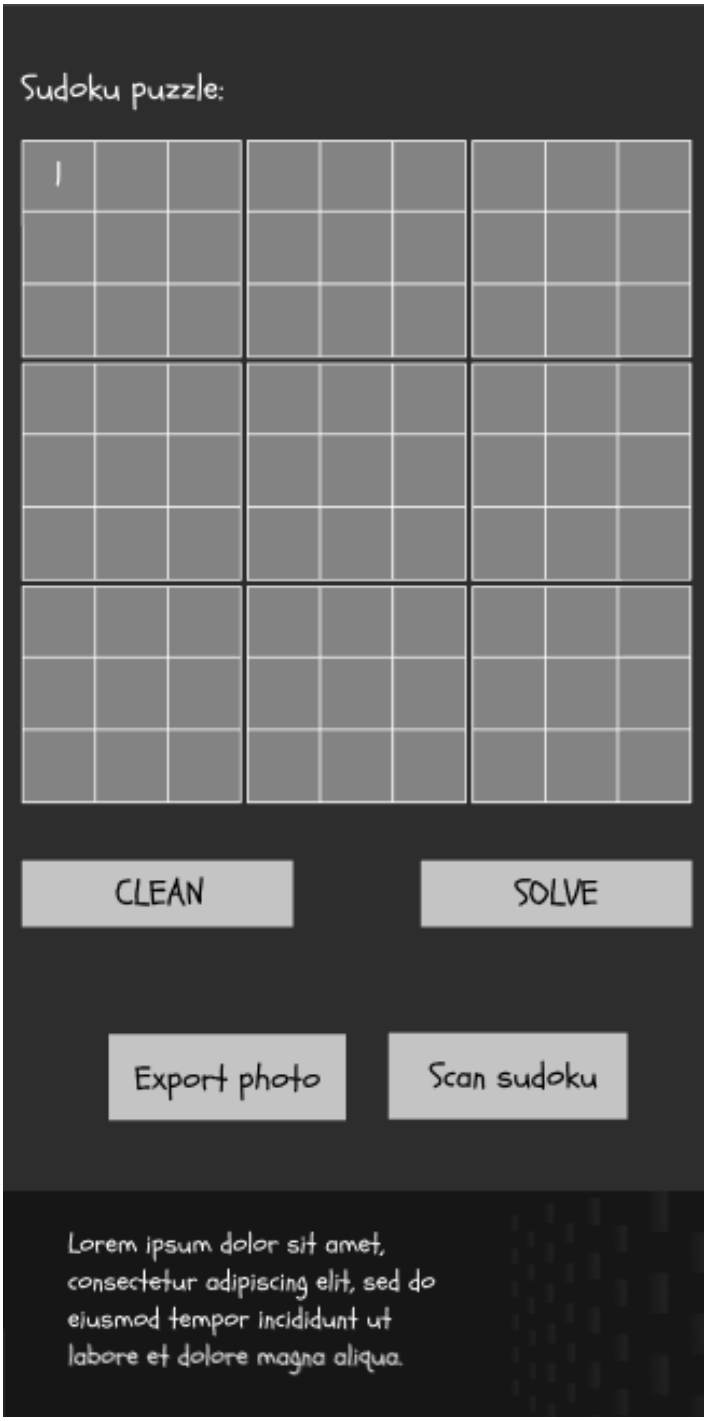

Рис 1.4. Прототип застосунку з усіма головними елементами [6]

[1] Цитата, яка була запропонована автором Sharyu [5]

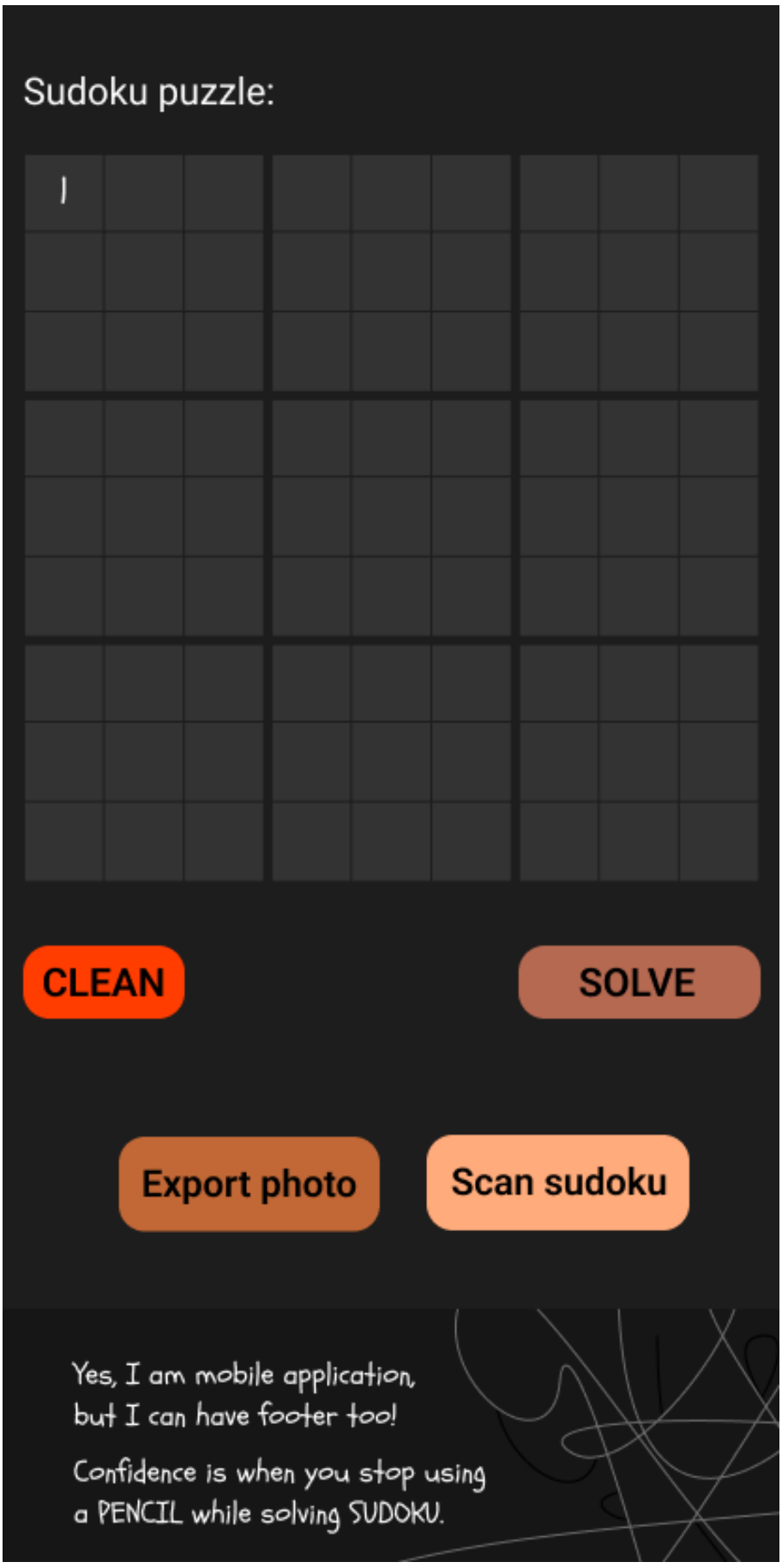

Рис 1.5. Кінцевий спрощений дизайн застосунку [6]

Програма має реалізувати вирішення «Судоку» з вводу користувача, файлу та камери, попереджати про помилки та давати змогу очистити введену або відображену інформацію про рішення з екрану.

Для досягнення поставленої мети необхідно:

- вивчити предметну галузь розв'язуваної задачі;
- проаналізувати усі можливі технології та алгоритми для виконання поставлених задач;
- обрати технології, мову та архітектуру застосунку.

1.5 Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик.

Вимоги до виконуваних функцій:

- Вирішення головоломки:
  - з вводу користувача;
  - з файлу зображення;
  - з камери.
- Очищення полів, які використовуються для введення з клавіатури або для відображення рішення.
- Рішення за допомогою експорту фото повинно спочатку відобразити зчитані числа на полі, а потім вирішити за командою користувача.
- Усі помилки пов’язані з введенням повинні відображатися спливаючому вікні та сповіщати користувача про місце помилки.
- Перемикатися між клітинами за допомогою клавіши «Tab» має бути доступним.

Організація вхідних даних:

- Числа з клавіатури;
- Зображення з носія;
- Камера на носії.

Організація вихідних даних:

– На екрані у клітинки для введення.
– На зображені з камери при натисканні кнопки «сканувати».

Обмеження по часу виконання: немає жодних обмежень у часі. У майбутньому можуть бути додані, у наслідок розвитку додатку.

Вимоги до мовного функціоналу: увесь функціонал у застосунку повинен бути англійською мовою. У подальшому додати можливість обирати одну з декількох мов.

Вимоги до інтерфейсу: реалізувати мінімальний інтерфейс для роботи із застосунком, у разі затвердження застосунку замовником та реалізувати повний дизайн з усіма додатковими можливостями:

– Зміна мови;
– Зміна теми інтерфейсу;
– Інструкція по використанню;
– Виділення рішення окремо від введених даних кольором;
– Додати можливість вирішення у режимі реального часу;
– Покращити якість зчитування з зображення.

1.5.2. Вимоги до інформаційної безпеки.

Вимог до інформаційної безпеки нема через те, що розроблена система у ході кваліфікаційної роботи не зберігає жодної приватної інформації про користувачів, яка повинна бути зашифрована та захищена від хакерського нападу. З боку контролю вхідної інформації застосунок повинен попереджати користувача про некоректність вводу або не давати змоги ввести недопустиме значення.

Час відновлення роботи застосунку має бути миттєвий після перезавантаження застосунку, а час очікування відповіді застосунку не повинен перевищувати трьох хвилин. Захистом від копіювання ПЗ будуть відсилки у коді

та розповсюдження вже компільованого додатку, без перенесення повного оригіналу, тільки моделі та зображення, які необхідні для правильного відображення інтерфейсу.

1.5.3. Вимоги до складу та параметрів технічних засобів.

Для роботи застосунку необхідний будь-який комп’ютерний або мобільний пристрій, який має такі технічні характеристики:

- Процесор 1 Gz;
- 1 Gb оперативної пам’яті;
- 1 Gb вільного місця на жорсткому диску;
- Операційна система:
    - Windows 10;
    - Linux (any);
    - Android 7.0;
    - IOS 13.0;
- Доступ до мережі інтернет на етапі встановлення застосунку або доступ до носія, з якого можна перенести застосунок;
- Повністю встановлений застосунок з усіма залежностями від сторонніх бібліотек;
- Доступ до виведення інформації – будь-який екран, який має здатність до відображення візуальної інформації;
- Доступ до введення інформації (сенсорний екран, клавіатура та миша або миша та здатність увімкнути екранну клавіатуру);
- Доступ до камери.

Наведені технічні характеристики засобів є лише рекомендаціями задля функціюванню відповідно до вимог щодо надійності, швидкості та безпеки, які були зазначені та узгодженні попередньо.

1.5.4. Вимоги до інформаційної та програмної сумісності.

Додаток повинен бути сумісним як з мобільними пристроями, так і з персональними комп’ютерами. Іншими словами, він повинен бути кроссплатформеним застосунком, який не вимагає додаткових налаштувань з боку користувача.

Будь-яка інформаційна сумісність відсутня, програма не має автоматизованої інформаційної системи, яка б зберігала та здійснювала обмін даними між застосунком та базою даних. Через це єдина вимога до інформаційної сумісності – можливість зчитувати зображення з файлів або з камери на носії.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

За завданням кваліфікаційної роботи було реалізовано розробку мобільного додатку для вирішення пазлу «Судоку».

Призначення розробленої системи:

- полегшити вирішення головоломки новачкам;
- забезпечити вирішення за допомогою комп'ютерного зору.

Розроблена програма реалізує наступні функції:

- Вирішення головоломки:
  - з вводу користувача;
  - з файлу зображення;
  - з камери.
- Очищення полів, які використовуються для введення з клавіатури або для відображення рішення.
- Рішення за допомогою експорту фото повинно спочатку відобразити зчитані числа на полі, а потім вирішити за командою користувача.
- Усі помилки пов'язані з введенням повинні відображатися спливаючому вікні та сповіщати користувача про місце помилки.
- Перемикатися між клітинами за допомогою клавіши «Tab» має бути доступним.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- розрізнення чисел з картинок та камери;
- зчитування вхідних даних з файлів та камери;
- вирішення головоломки;
- сповіщення про помилки.

2.2. Опис застосованих математичних методів

Додаток використовує сторонню бібліотеку для вирішення головоломки (Рис 2.1). Ця бібліотека надає можливість генерувати головоломки та вирішувати їх. Для вирішення «Судоку» було використано алгоритм «пошук з повертанням», який є Brute-force алгоритмом, який послідовно перебирає сусідів клітинки. Якщо рішення не було знайдено, клітинка сусіда, яка була заповнена тим самим числом буде очищена, якщо це не було введено користувачем. Таким чином воно буде ходити у циклі намагаючись вирішити пазл.

```
def _solve(self, raising=False):
    blanks = self.__get_blanks()
    blank_count = len(blanks)
    are_blanks_filled = [False for _ in range(blank_count)]
    blank_fillers = self.__calculate_blank_cell_fillers(blanks)
    solution_board = self.__get_solution(
        Sudoku._copy_board(self.sudoku.board), blanks, blank_fillers, are_blanks_filled)
    solution_difficulty = 0
    if not solution_board:
        if raising:
            raise UnsolvableSudoku
        solution_board = Sudoku.empty(self.width, self.height).board
        solution_difficulty = -2
    return Sudoku(self.width, self.height, board=solution_board, difficulty=solution_difficulty)
```

Рис 2.1. Код вирішальника пазлу

У деяких випадках таке рішення потребує багато часу на вирішення, тому в подальшому можна оптимізувати алгоритм та використати деякі варіанти полегшення пошуку рішення.

Наприклад, серед варіантів полегшення роботи алгоритму:

– знайдення єдиної пустої клітинки у рядку;
– знайдення єдиної пустої клітинки у стовпці;
– знайдення єдиної пустої клітинки у блоці;
– використати алгоритм Кнута з танцюючими посиланнями.

Останній спосіб оптимізації алгоритму дуже цікавий та досить складний для розуміння. Танцуючі посилання (DLX)[7] – це метод додавання та видалення вузла з кругового двозв’язаного списку. Такий список ефективно працює для

алгоритмів зворотнього відстеження, яким є алгоритм X Дональда Кнута. Головоломка є типовим представником проблеми точкового покриття, а алгоритм Кнута спрямований на знаходження рішення за допомогою рекурсивності, недетерміновості алгоритму та його повернення у глибину.

В алгоритмі X рядки та стовпці регулярно вилучаються з матриці та відновлюються до неї. Виключення визначаються шляхом вибору стовпця та рядка в цьому стовпці. Якщо вибраний стовпець не містить рядків, поточна матриця нерозв'язна і її потрібно повернути назад. Коли відбувається видалення, усі стовпці, для яких обраний рядок містить число, а також усі рядки (включаючи вибраний рядок), які містять число у будь-якому з вилучених стовпців, видаляються. Стовпці видаляються, оскільки вони заповнені, а рядки видаляються, оскільки вони конфліктують із обраним рядком. Щоб видалити один стовпець, спочатку видаліть заголовок вибраного стовпця. Цей порядок гарантує, що будь-який видалений вузол буде видалено точно один раз і в передбачуваному порядку, щоб його можна було належним чином повернути. Якщо в отриманій матриці немає стовпців, то всі вони заповнені і виділені рядки утворюють рішення.

Щоб повернутися назад, вищезазначений процес потрібно повернути назад, використовуючи другу частину алгоритму зазначену вище.

Якщо казати на простому прикладі, то в нас є головоломка з кільцями та трьома палками, на яких розташовані такі кільця, які необхідно виставити у зростаючому порядку переставляючи їх послідовно.

Такий алгоритм працював би таким чином:

- спробувати переставити декілька кіл;
- якщо не можна переставити жодного кільця, то повертаємось на декілька кроків у зворотньому напрямку до того моменту, коли був вибір між двома переставленнями кіл;
- пробуємо переставити інше кільце;
- повторюємо, доки умови рішення не буде досягнуто.

2.3. Опис використаних технологій та мов програмування

Додаток повністю реалізовано на Python мові програмування з використанням фреймворку Kivy для інтерфейсної частини додатку, пакетів imutils для полегшення роботи з OpenCV, бібліотек TensorFlow, OpenCV, Scikit-image, SciPy для роботи з зображенням та бібліотеки py-sudoku для вирішення гри (всі залежності встановлені пакетним менеджером Pip [8]).

Мова Python у більшості випадків використовується для вирішення невеликих задач та є більше скриптовою або функційною мовою, аніж мовою для розробки інтерфейсу. Але вибір мови був визначений темою роботи [9] – саме завдяки кількості вже готових рішень у бібліотеках не потрібно вигадувати «велосипед». Усе необхідне вже існує, його необхідно лише встановити та скомпонувати.

Однак мова Python не є типовою для мобільних додатків, оскільки на цьому сегменті укорінилися інші мови такі як Java, наприклад. Але це не значить, що Python не підходить для розробки мобільних застосунків. Також ця мова дуже зарекомендувала себе за рахунок великої ефективності у сферах Machine Learning та Data Science. Її розширюваність та гнучкість дозволяє дуже легко оперувати даними та один із слоганів цієї мови «Just Import!» - перш ніж розробляти щось перевір чи не вирішили вже цю проблему хтось інший. Будь-яке завдання або потреба може вже мати окремий фреймворк або бібліотеку для вирішення.

Інтерпретованість Python надає можливість реалізувати кроссплатформеність напряму «з коробки», адже інтерпретатор існує для багатьох популярних платформ та встановлюється за замовчуванням на більшість дистрибутивів Linux.

Вибір бібліотек обумовлено їх функційністю, популярністю та ефективністю. OpenCV – це набір типів даних, функцій та класів для обробки зображень алгоритмами комп'ютерного зору. Вона абсолютно безкоштовна та підходить для виконання поставленої задачі з визначення чисел на зображеннях.

OpenCV обробляє зображення та «витягує» лише ті дані, які необхідні для розпізнавання та передає їх на наступний крок обробки.

Додатково було використано бібліотеку Scikit-image для обробки зображення, OpenCV має такі самі методи обробки, але Scikit-image у деяких моментах надає більше можливостей у обробці.

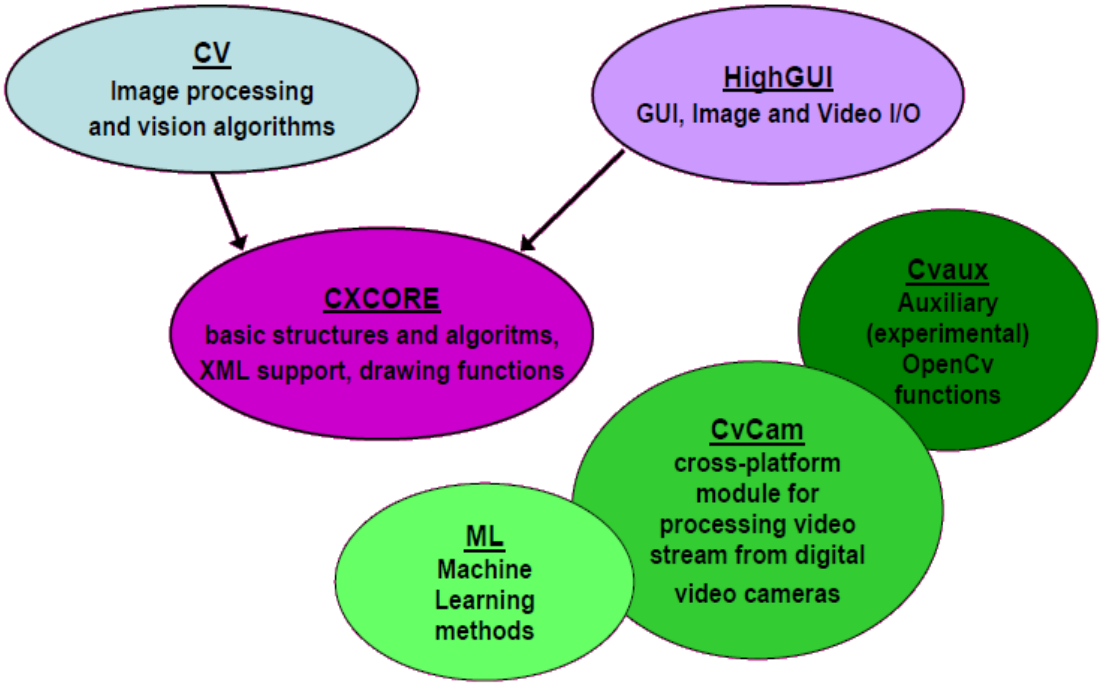

Рис 2.2. Основні модулі бібліотеки OpenCV

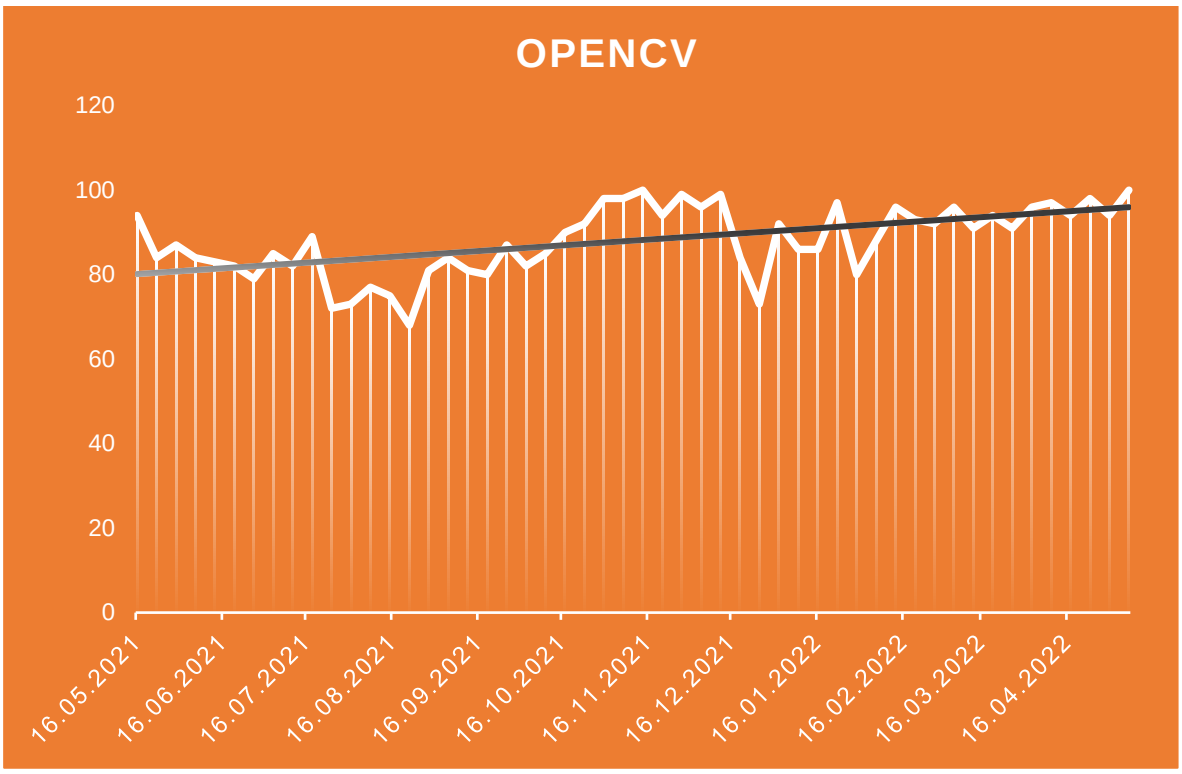

Рис 2.3. Популярність запитів «cv2» у Google за останній рік [2]

TensorFlow [10] – інструмент, який вирішує питання навчання програми розпізнавати числа. Вона полегшує процес створення нейронної мережі та надає вже заготовлені масиви даних (datasets) для навчання програми відрізняти цифри на зображеннях. Кожне зображення – це масив пікселів (тензор), потік яких необхідно обробити. Щоб визначити що саме є на зображені необхідно надати програмі приклади того, що вона повинна знаходити, з ключами, які вона повертатиме, якщо знайде співпадіння. TensorFlow порівнює зображення та намагається з якомога більшою точністю виявити об’єкт на зображені. Головне не перевчити систему – тоді може статися так, що у разі, якщо програма не буде на дев’яносто дев’ять і дев’ять відсотків впевнена у тому, що зображено, вона не «скаже» про те, що «бачить».

Нещодавно його популярність зменшилась через інший фреймворк глибокого навчання, PyTorch, який було розроблено командою Facebook і відкрито на GitHub у 2017 році.

Таб 2.1 Таблиця порівняння фреймворків

Фреймворк	Плюси	Мінуси
PyTorch	+ Кодування, подібне до Python. + Динамічний графік. + Легке та швидке редагування. + Хороша документація та підтримка громади. + Відкрите джерело. + Популярність.	– Для візуалізації потрібна стороння сторона. – API-сервер, необхідний для виробництва.
TensorFlow	+ Простий вбудований високорівневий API. + Візуалізація навчання за допомогою Tensorboard. + Придатний до виробництва. + Проста мобільна підтримка. + Відкрите джерело. + Добре зроблена документація.	– Статичний графік. – Метод налагодження. – Важко внести швидкі зміни.

З таблиці наведеній вище було зроблено висновки про вибір фреймворку. Очевидно, що для вирішення задач, які поставлені для цього проекту, більше підходить TensorFlow, як більш надійне рішення для мобільного застосунку з більш простим залученням його у програму. Цей фреймворк вирішує проблеми візуалізації, готовності до виходу на ринок, а також сумісність з мобільними пристроями.

SciPy бібліотеку було використано у якості інструменту для виведення рішення на зображення. Оскільки користувач буде використовувати камеру, то необхідно надати можливість відображення рішення поверх зображення – це знадобиться в майбутньому, якщо буде затверджено початок додання вирішення у режимі реального часу.

Kivy – це безкоштовний та відкритий фреймворк для розробки мобільних застосунків з підтримкою NUI інтерфейсу – сенсорного екрану, мультитач дій, голосових допоможників. Він підтримує платформи Android , iOS , Linux , macOS та Windows. Окрім цього фреймворк працює з графічною бібліотекою, яка використовує OpenGL та заснована на Vertex Buffer Object й шейдерах. Це надає можливість міграції у веб-простір.

Kivy має проміжну мову завдяки якій можна створювати користувацькі віджети. Ця мова дуже схожа на суміш CSS та HTML. На CSS вона схожа стилями, параметрами, які вказуються до елементів щоб відобразити їх на екрані, а на HTML – деревовидною структурою. Кожен віджет може бути вкладений у інший віджет.

Окрім цього цей фреймворк рекомендовано для початкових проектів. Один із розробників проекту PyMT, з розробки модуля мультимедійних застосунків з підтримкою мультитач, Пассей Дирето, посилався в статті на Kivy фреймворк [11]: «We have performed numerous benchmarks and as it turns out, to achieve the great speed and flexibility that Kivy has, we had to rewrite quite a big portion of the codebase, making this a backwards-incompatible but future-proof decision. Most notable are the performance increases, which are just incredible.». Що у перекладі звучить так: «Ми провели численні тести, і, як виявилося, щоб досягти високої швидкості та гнучкості, якими володіє Kivy, нам довелося переписати досить велику частину кодової бази, зробивши це рішення несумісним, але надійним у майбутньому. Найбільш помітним є збільшення продуктивності, яке просто неймовірне.»

Є інший фреймворк, який потенційно можна було використати, але після аналізу було вирішено використовувати саме Kivy. Цей фреймворк PyQt [12], який більше підходить до десктопних застосунків, аніж до мобільних. Однак, це не лише єдиний мінус, який вплинув на остаточне рішення щодо графічного фреймворку.

Kivy – це стовідсотково безкоштовний фреймворк, який має ліцензію Массачусетського технологічного інституту, що дає використовувати його при розробці комерційних продуктів. PyQt, навпаки, підходить лише для некомерційних проектів.

Беручи до уваги, що PyQt не такий зручний для користувача, оскільки Qt не є якоюсь мовою, він використовує C++ як мову програмування, що трохи ускладнює розробнику розробку інтерфейсу користувача, вирішення щодо фреймворку було достатньо легким.

Приклад використання проміжної мови KV від Kivy наведено на Рис 2.4.

```
<ErrorPopup>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        Label:
            text: root.information
            text_size: self.size
            font_size: '14sp'
            halign: 'left'
            valign: 'top'
            pos: (10, 20)
            size_hint_y: 0.5
        Button:
            size_hint: None, None
            size: 80, 30
            text: "OK"
            pos: (1000, 0)
            on_release: root.ok(root.popup)
```

Рис 2.4. Приклад використання проміжної мови KV

2.4. Опис структури системи та алгоритмів її функціонування

Структура додатку – це описання маршруту користувача, як він мандрує серед інтерфейсу. Чим простіше розроблений цей шлях – тим більше буде

зацікавленість клієнта. Застосунок не має багато сторінок, лише три, які зображені на Рис 2.5.

Головне вікно представляє собою набір елементів для переходу на інші сторінки та введення даних з клавіатури.

Файл менеджер відкриває додаткове вікно перегляду файлової структури на носії. З нього можна обрати файл для аналізу на предмет головоломки «Судоку».

Вікно камери передбачає відкриття камери та схоплення зображення для аналізу та вирішення поверх нього пазлу. Також на цьому вікні є елемент ля повертання на головне вікно.

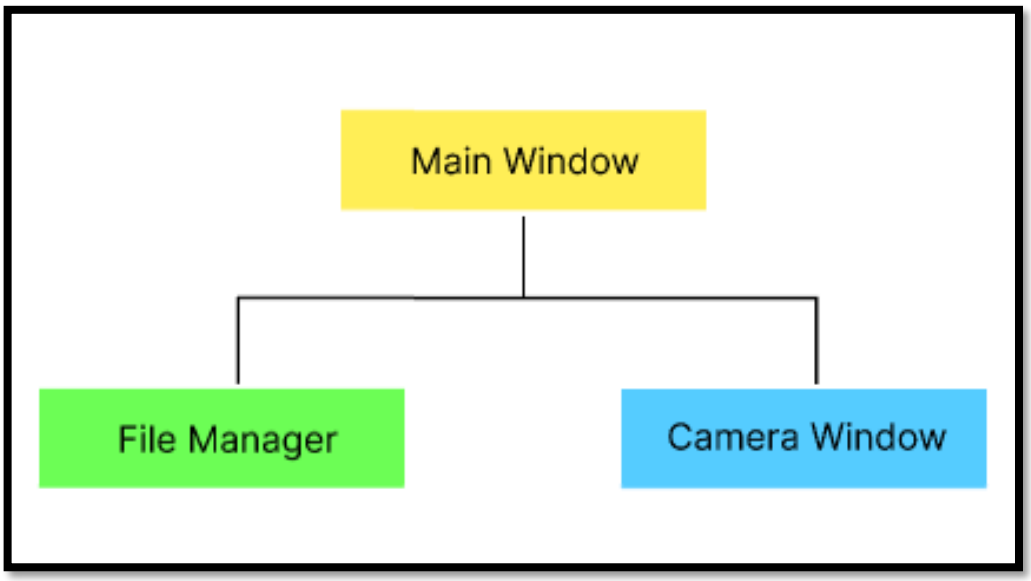

Рис 2.5. Структура системи [6]

З точки зору користувача було б доречно додати елемент, який би відображав завантаження (Рис 2.6). Коли програма вирішує головоломку користувач не розуміє чи працює програма, чи вона не відповідає на запити. Для цього при будь-яких великих очікуваннях, які перевищують 5 секунд, необхідно додати елемент перекриваючий вікна або додаткове вікно завантаження.

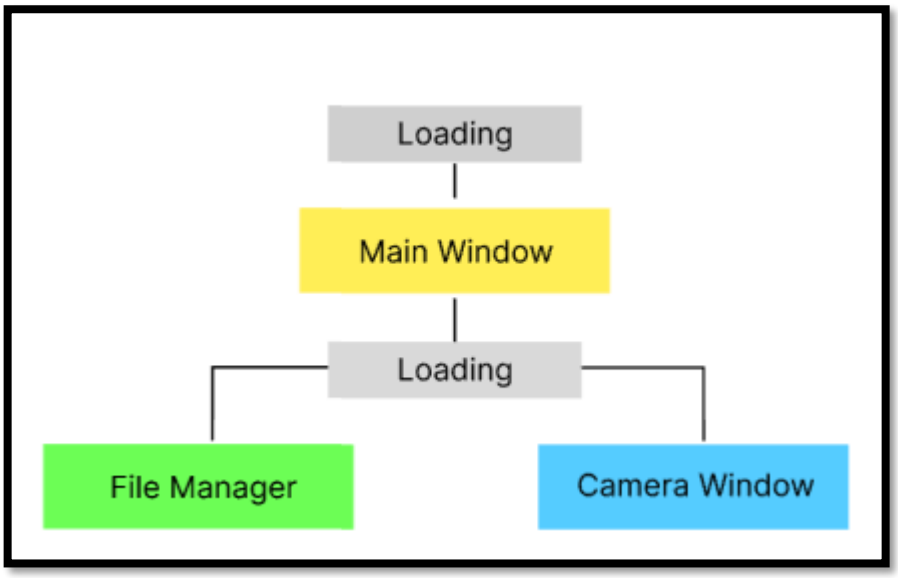

Рис 2.6. Бажана структура системи [6]

Існує два варіанти кінцевого результату взаємодії користувача із застосунком. Перший – помилка введених даних, другий – виведення рішення. Обидва можуть бути досягнені з будь-якої опції по введенню даних користувачем (Рис 2.7, Рис 2.8).

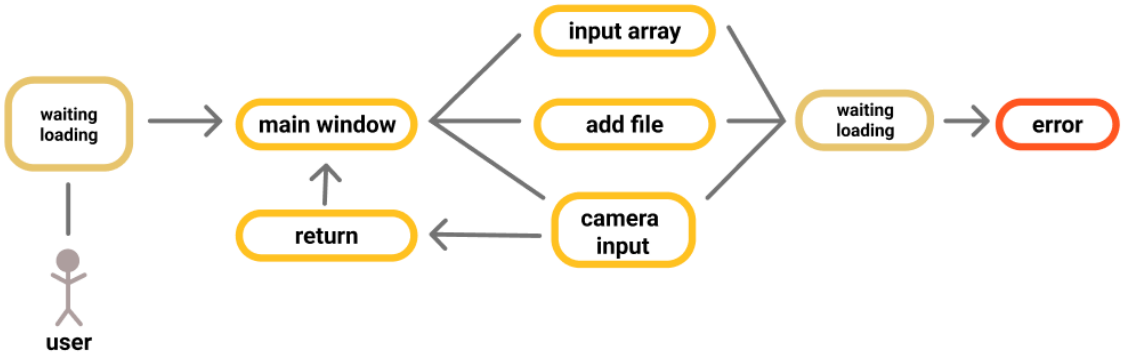

Рис 2.7. Варіант розвитку взаємодії з кінцевим результатом помилки [6]

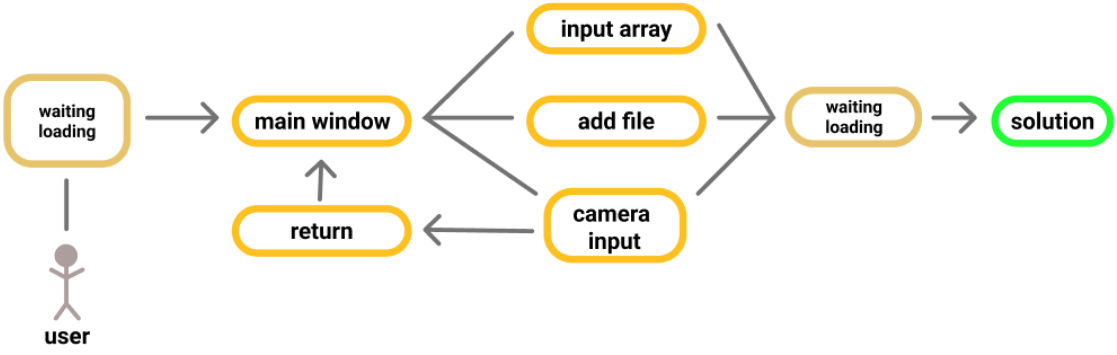

Рис 2.8. Варіант розвитку взаємодії з кінцевим результатом рішення [6]

Помилка не призводить до завершенню програми, а лише попереджує користувача про некоректність вводу та неможливість вирішити головоломку з наданими даними.

Для створення додатку було використано адаптований під потреби проекту шаблон проектування для створення архітектури застосунку MVC. Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») – це дуже популярний патерн, який використовується для багатьох рішень щодо організації файлів, логіки та компонентів. Таким чином архітектура та робота програми стає більш прозорою, знайомою та легшою для сприйняття.

Цей шаблон було адаптовано до мети кваліфікаційної роботи (Рис 2.9). Модель – генерується лише один раз та являє собою набір даних для розпізнавання зображень. Контролером є аналізатор, який знаходить рішення за допомогою алгоритмів та звернення до моделі. Представлення – це інтерфейс з компонентами, до яких звертається користувач, а він у свою чергу звертається до аналізатора, який виконує необхідну дію використовуючи модель даних.

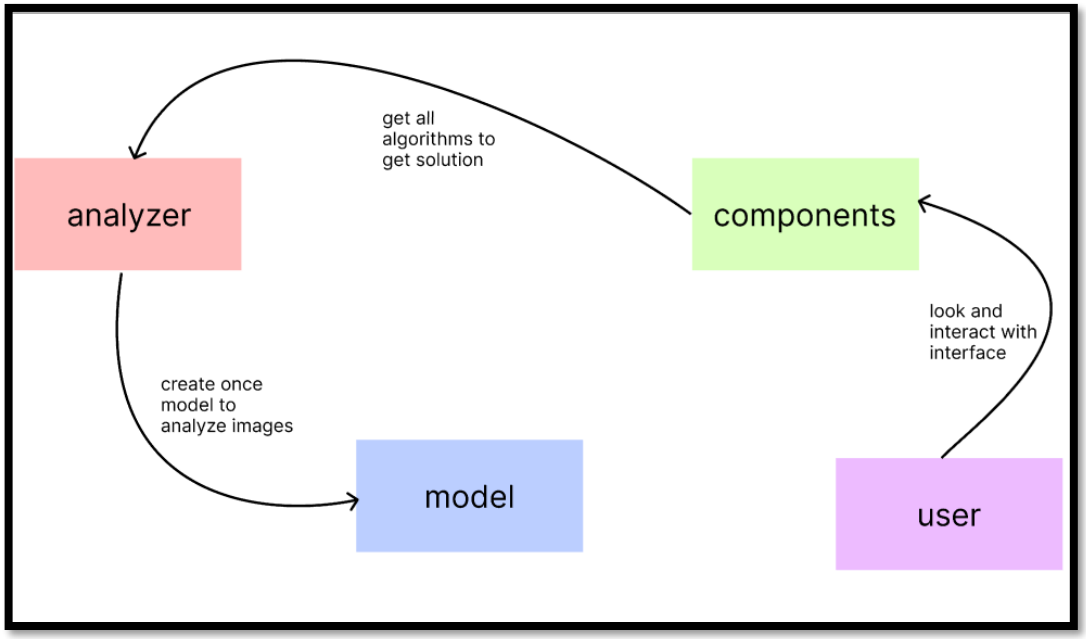

Рис 2.9. Схема взаємодії компонентів додатку [6]

Модель представлена у вигляді файлу з розрішенням .h5, який генерується один раз, проходячись протягом 10 епох по заготовленому масиву даних для

навчання мережі, досягаючи точності розпізнавання у більше ніж дев’яносто відсотків.

Зручність Python полягає у тому, що можна створити окреме ізольоване оточення (Рис 2.10), яке буде мати в собі усі необхідні залежності та бібліотеки. Інколи буває, що декілька проектів потребують різні версії бібліотек або деякі бібліотеки мають конфлікти через однакову назву. Тому все це зберігається в окремій папці та коли необхідно запускається через консоль.

Рис 2.10. Віртуальне оточення

Стосовно розміщення файлів у директоріях – файли рознесені по окремим папкам в залежності до їх призначення (Рис 2.11). Наприклад:

- папка аналізатора містить у собі файли пов’язані з обробкою та аналізом даних;
- папка камери містить у собі компоненти для відображення камери;
- компоненти – усі віджети та елементи для відображення інтерфейсу;
- швидкі скрипти має у собі скрипти для Windows cmd для зручного налаштування та відкриття проекту;
- картинки – зображення, які потрібні для роботи застосунку;
- модель містить у собі файли для навчання та створення моделі.

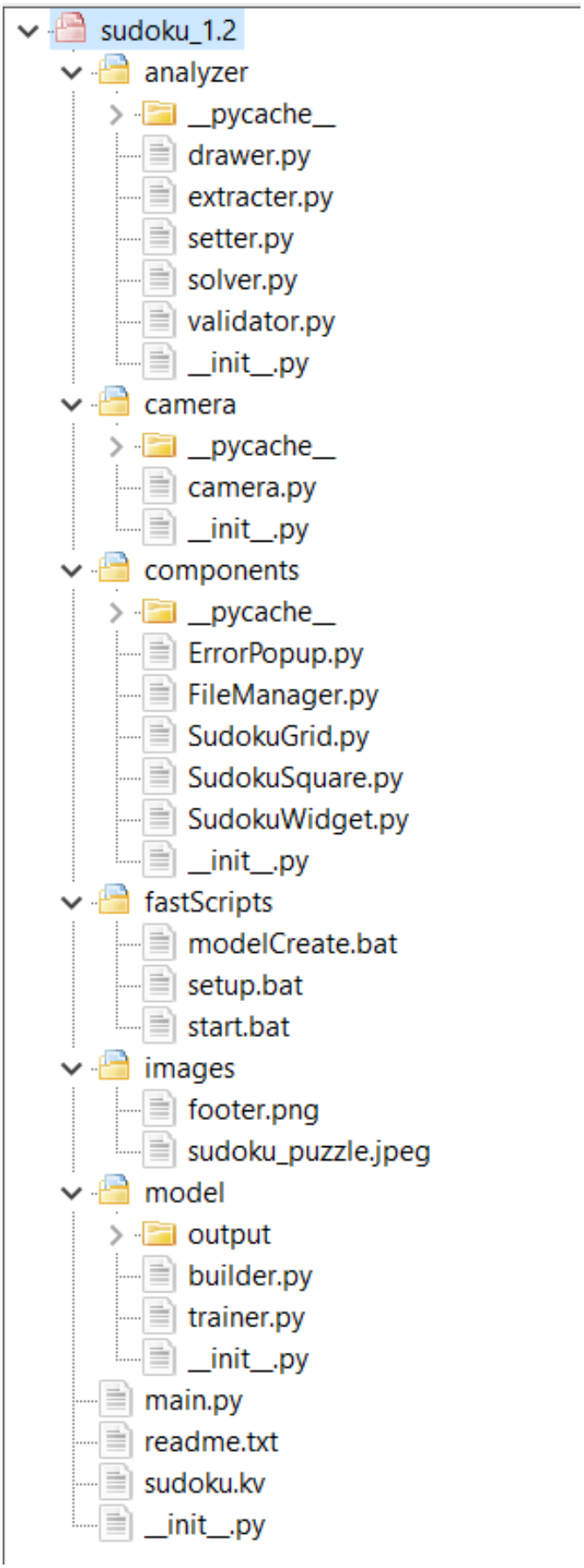

Рис 2.11. Структура файлової системи

Також для гарної співпраці з іншими розробниками та збереження даних видалено необхідно додати весь проект до видаленого репозиторію GitHub.

Гарною практикою є використання систем контролю версій під час розробки додатку. Це допомагає слідити за етапами розробки та у разі якихось прогалин або дефектів швидко їх вирішувати. Тому для цього проекту було використано VCS Git, яка також має власний видалений репозиторій для проектів.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом введення їх користувачем, завантаження інформації із файлу або з камери пристрою.

Вхідні дані:

- числа з масиву;
- зображення.

Вихідні дані:

- масив 9 на 9 з числами;
- зображення з даними поверх нього.

Файли зберігаються на пристрої або під час відкриття програми. Після закриття програми усі дані, яких не існувало до цього, не будуть збережені

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Пристрій, на якому проводилось тестування:

- процесор Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz, 1498 Mhz, 4 Core(s), 8 Logical Processor(s);

– операційна система Windows 10 Enterprise;
– оперативна пам’ять 16.0 GB;
– наявна система охолодження;
– екран з розширенням 1920 x 1080 x 60 hertz;
– доступ до мережі Internet;
– маніпулятор "миша";
– клавіатура.

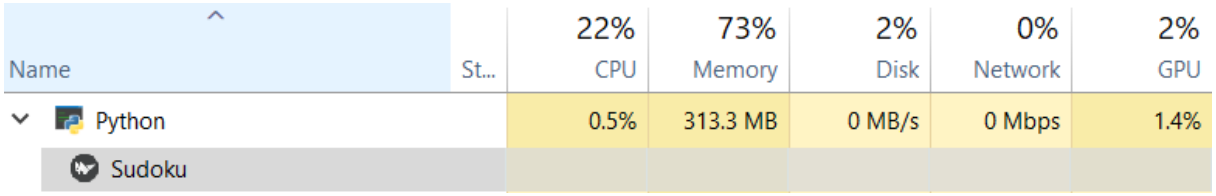

Рис 2.12. Технічні заміри працюючого застосунку

Наведені вище технічні характеристики є рекомендованими, тобто розробник не може гарантувати, що розроблений програмний додаток буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки так само і на інших пристроях.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування Python з використанням фреймворку Kivy, пакетів imutils, бібліотек TensorFlow, OpenCV, Scikit-image, SciPy та py-sudoku.

В якості IDE для розробки був використано Notepad++. Для встановки усіх залежностей було долучено систему управління пакетами для Python – Pip.

Необхідними програмними та технічними засобами для клієнта є мобільний телефон, ПК або інший девайс з підключенням до Інтернету для завантаження додатку.

2.6.3. Виклик та завантаження програми

Для виклику та завантаження необхідно виконати наступні дії:

– необхідно встановити усі бібліотеки, пакети та фреймворки;

– згенерувати файл з моделлю (необов’язково);

– запустити через командну строку або через скомпонований файл застосунок.

2.6.4. Опис інтерфейсу користувача

Користувач може ввести дані за допомогою клавіатури послідовно до кожної клітинки, переключаючись між ними або за допомогою миші або натискаючи на клавішу «Tab».

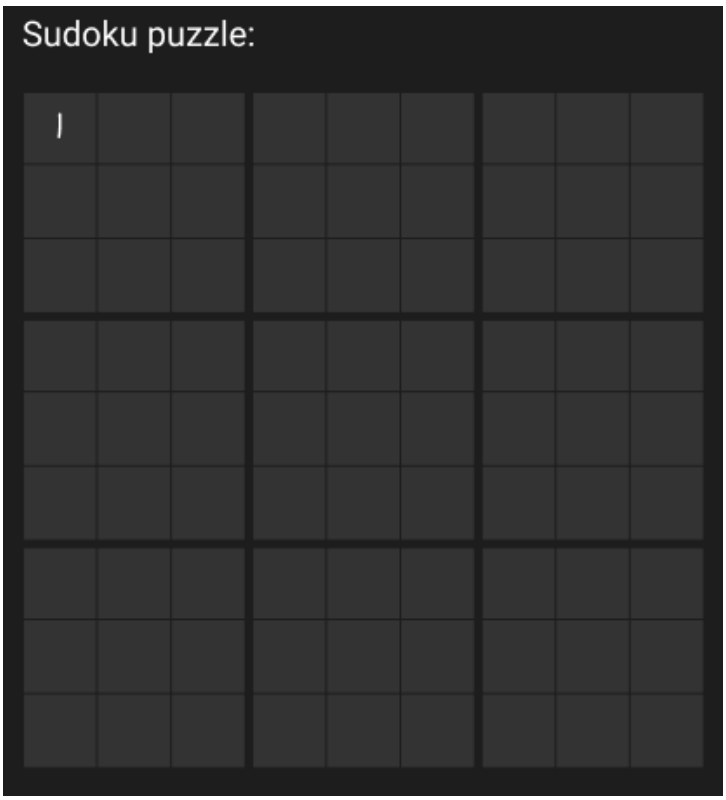

Рис 2.13. Поле 9 на 9 [6]

Потім користувач може натиснути на кнопку «Вирішити», щоб отримати рішення у тому ж самому полі дев’ять на дев’ять.

Рис 2.14. Кнопка «Вирішити» [6]

Якщо користувач припустився помилки, він може настиснути на «Backspace» або на кнопку «Очистити», якщо він хоче очистити усі клітинки та ввести знов нові дані.

Рис 2.15. Кнопка «Очистити» [6]

Якщо користувач хоче обрати файл з носія для обробки та вирішення з нього, він повинен натиснути на кнопку «Експортувати фото».

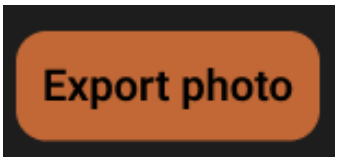

Рис 2.16. Кнопка «Експортувати фото» [6]

Якщо користувач хоче зчитати зображення з камери, то він може натиснути кнопку «Сканувати Судоку». Тоді відкриється нове вікно з ввімкненою камерою.

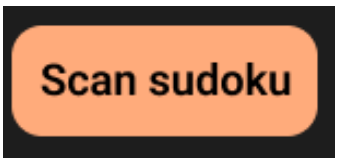

Рис 2.17. Кнопка «Сканувати Судоку» [6]

Футер не несе жодної функційної або інформаційної складової для застосунку, але його було додано для підняття настрою користувача.

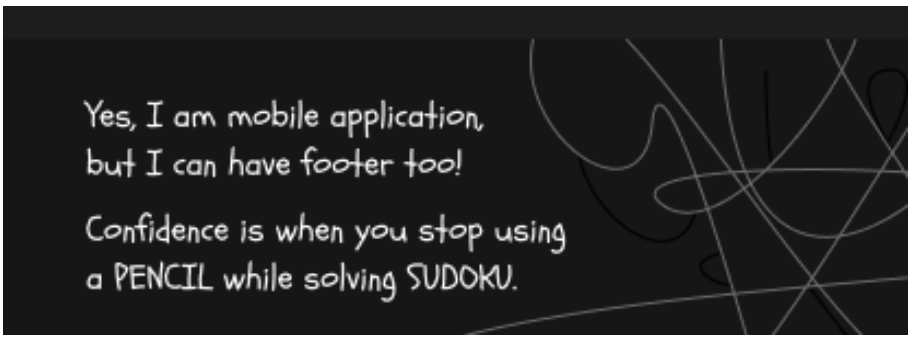

Рис 2.18. Футер [6]

Якщо казати про майбутній інтерфейс застосунку, то для цього було зроблено макет у вигляді презентації, де розділено кожен елемент та його роль. Розглянемо його більш детально та зробимо висновок чому саме цей макет було відкладено на майбутнє.

Рис 2.19. Основні елементи застосунку [6]

На Рис 2.19 зображені всі елементи, які використовуються у застосунку – камера, поле з судоку, шапка, повідомлення про помилку, кнопка повертання назад, іконка додатку, кнопка видалення, кнопка інформації про застосунок, кнопки вирішення судоку, відкриття камери та файлу. Для цих елементів було використано 5 кольорів – два темних, білий, помаранчевий та червоний. Останній необхідно було використати для акценту на дію кнопки, щоб користувач не переплутав з кнопкою вирішення пазлу.

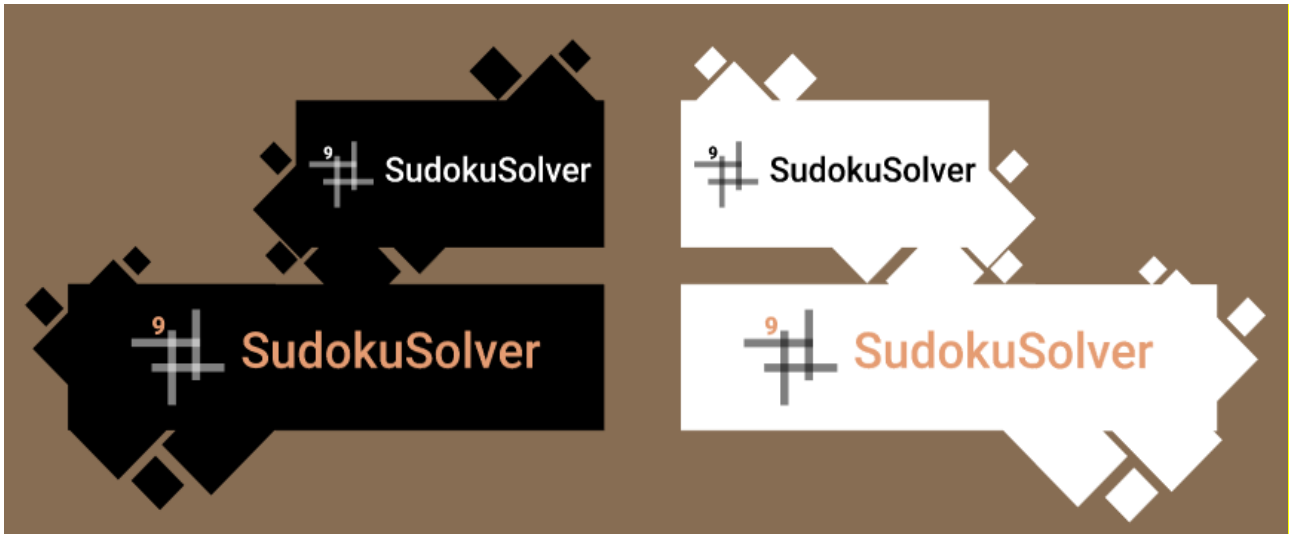

Рис 2.20. Логотип додатку [6]

Для логотипу (Рис 2.20) застосунку було використано ключові асоціації з грою судоку – перехрестя ліній, які створюють клітинки; дев’ять клітинок по дев’ять блоків; цифра дев’ять; назва гри. Логотип було представлено у чотирьох варіантах. Перший – повністю білий, другий – білий з акцентами помаранчевим кольором на цифру дев’ять та назву застосунку, третій – повністю чорний та четвертий – чорний, також з акцентами помаранчевим кольором на цифру дев’ять та назву застосунку.

Для блоку з презентацією логотипів було використано такий опис додатку (Рис 2.21): «Судоку Вирішувач. Головні елементи цієї гри – лінії та цифри. Стиль та почуття цієї гри це мінімалізм та красота чисел. Помаранчевий означає веселість, креативність, успіх, щастя, забаву – судоку це перш за все гра и вона повинна відчуватися веселою».

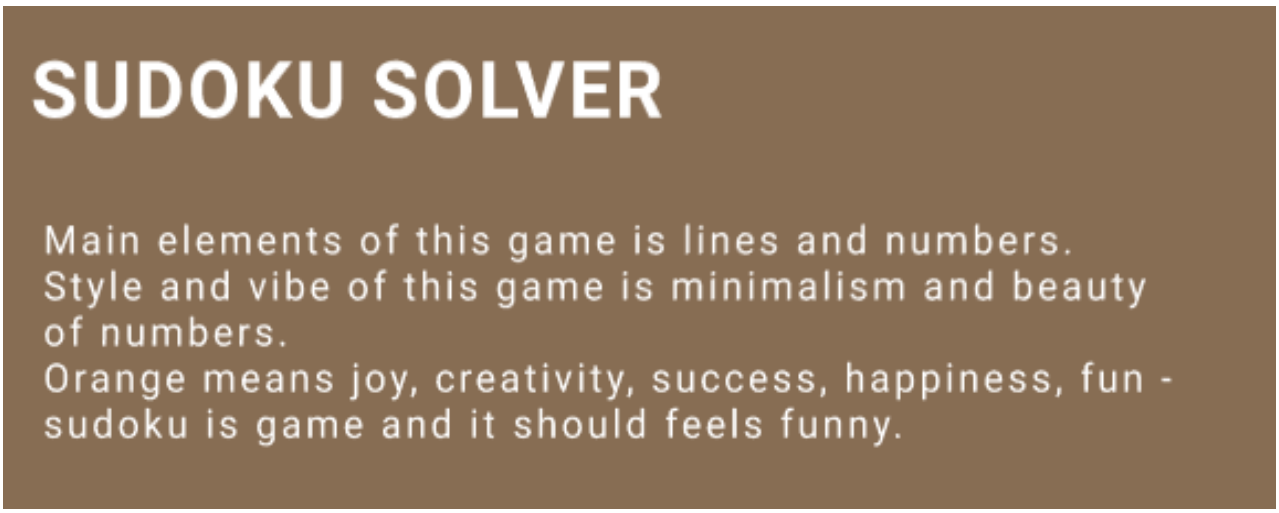

Рис 2.21. Опис додатку у презентації дизайну логотипу [6]

Було розроблено декілька мокапів того, як повинні виглядати основні варіанти розвитку програми при взаємодії з користувачем. Їх можна розділити на варіанти з помилками та роботи у звичайному режимі. Проаналізуємо спочатку роботу без помилок або попереджень.

Всього таких макетів три (Рис 2.22, Рис 2.23, Рис 2.24, Рис 2.25) – на них відображено режим роботи з камерою та введення з клавіатури користувачем.

На дизайні введення з клавіатури (Рис 2.23) присутні такі елементи як поле дев’ять на дев’ять для введення чисел, шапка з логотипом у біло-помаранчевому кольорі, кнопка очищення поля, кнопка для отримання ришення, кнопки додаткових режимів введення даних – з файлу та з камери.

Рис 2.22. Мокапи роботи у звичайному режимі [6]

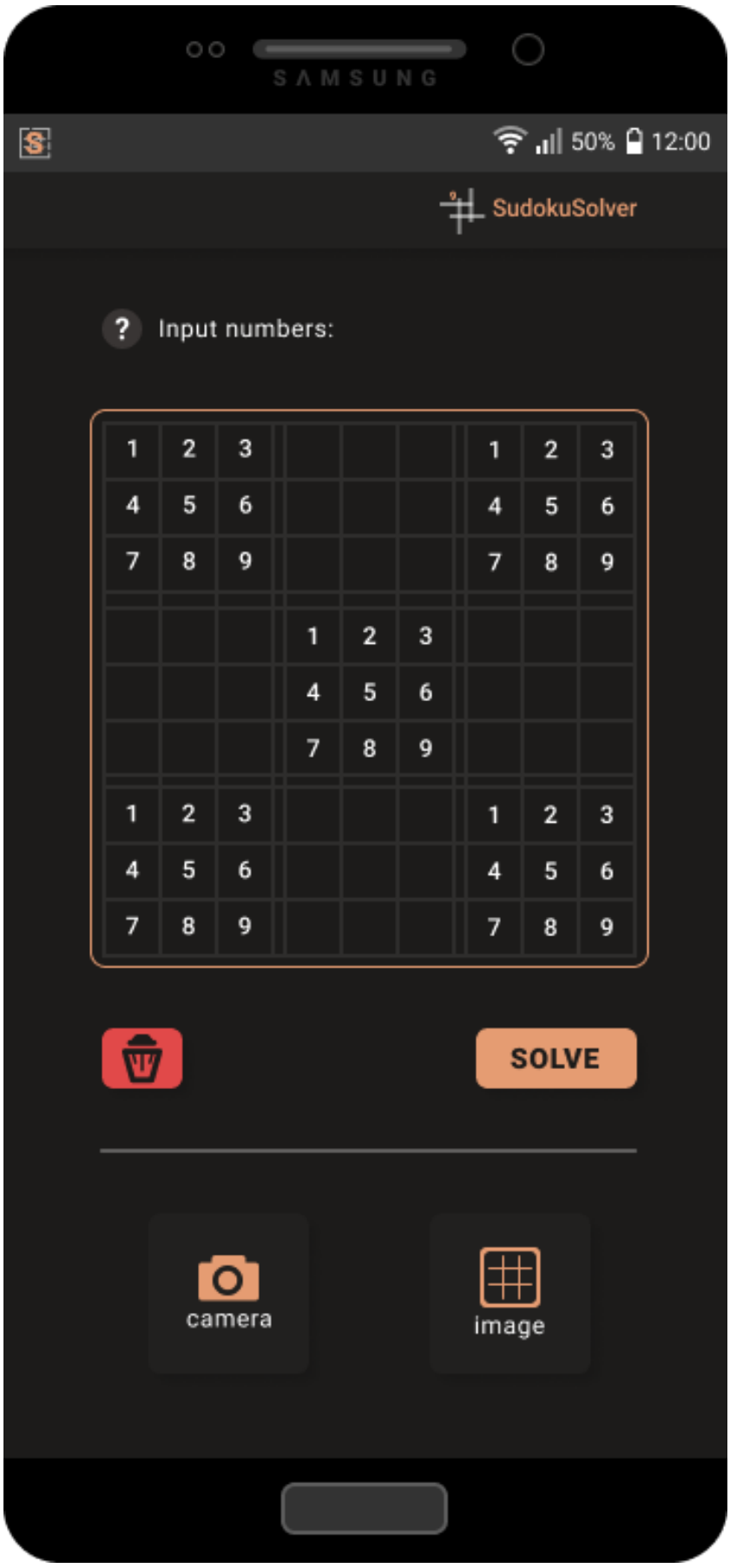

Рис 2.23. Макет введення з клавіатури [6]

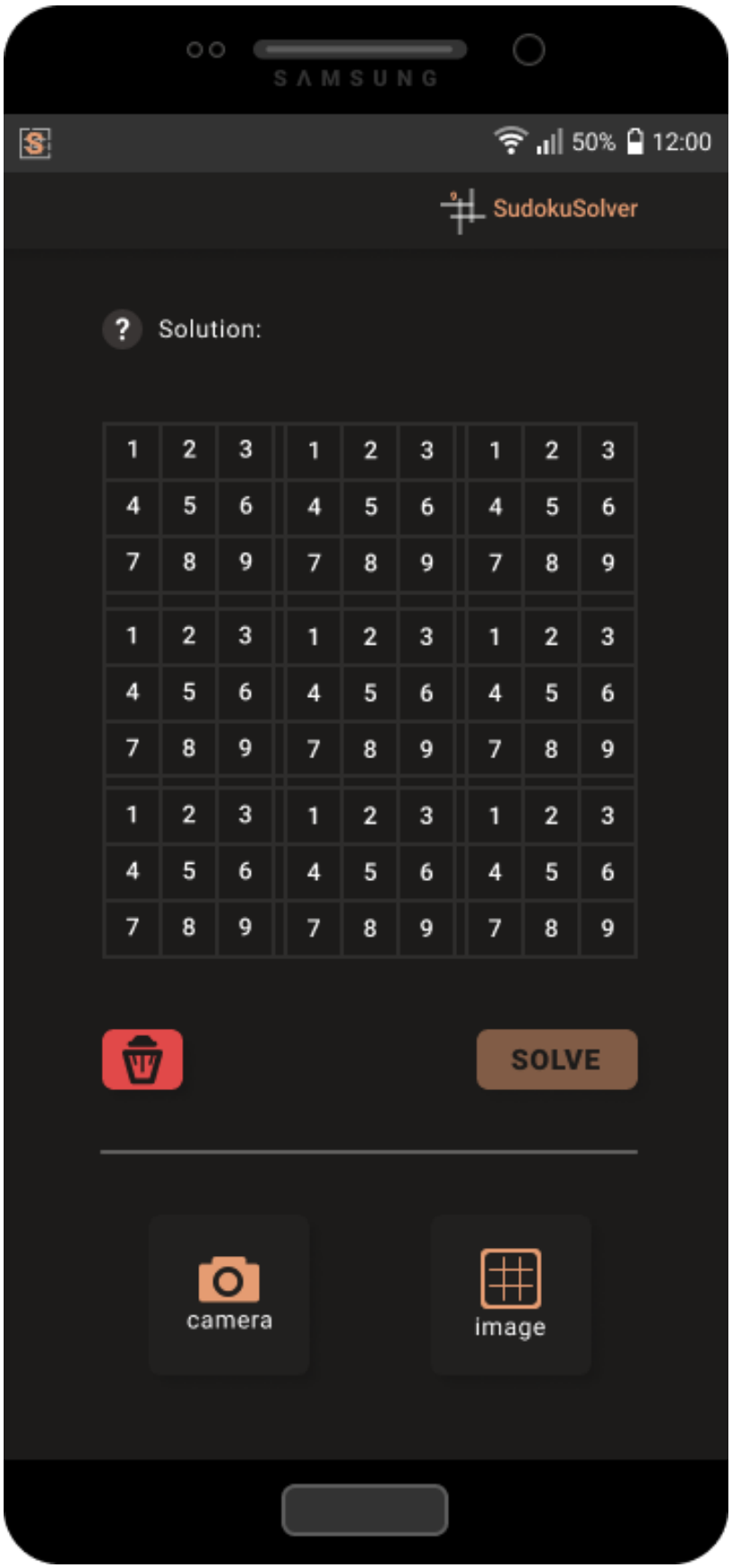

Рис 2.24. Макет введення з клавіатури з вирішеним судоку [6]

Макет з вирішеною головоломкою (Рис 2.24) не відрізняється нічим від попереднього окрим того, як повинно виглядати поле після вирішення. Кнопка вирішення становиться неактивною.

Для режиму введення з камери (Рис 2.25) також було створено дизайн. Для цього було розміщено поле для камери майже на увесь екран, адже це основний елемент з яким буде взаємодіяти користувач. Необхідно надати якомога більше місця для відображення зображення з камери. На цьому полі розміщено умовні позначки – таким чином користувач розуміє як краще розмістити судоку під камерою, щоб досягнути кращого результату у зчитуванні чисел. Чим краще зображення, тим більше ймовірність того, що усі числа будуть зчитані, а рішення буде вірним.

Щоб піймати момент, коли зображення має найбільшу якість є кнопка вирішення – вона припиняє зчитування з камери та «фрізить» останнє, що вона побачила. Для повертання до головного екрану є кнопка зі стрілкою уліво, яка розуміється, як повернення назад.

Також треба звернути увагу, що, коли застосунок працює, зверху відображається іконка додатку – таким чином користувач сповіщається про роботу застосунку, його помилки та повідомлення.

Рис 2.25. Макет з камерою [6]

Дизайн попереджень (Рис 2.26, Рис 2.27, Рис 2.28, Рис 2.29, Рис 2.30) дуже схожі між собою, єдина різниця у тексті та екрані, на якому воно буде відображатися.

Основні дві помилки – це зчитати зображення та знайти рішення. Тому перше повідомлення звучить як «Вибачте, ми не змогли знайти рішення», а друге – «Вибачте, ми не змогли зчитати зображення».

Перше повідомлення з’являється, якщо не можна знайти рішення пазлу, а друге – якщо не вірний тип файлу було обрано або судоку не пройшло валидацію з зображення. Також необхідно створити інше попередження про те де саме невірно знайдено число, щоб конкретизувати в чому була саме помилка.

Рис 2.26. Мокапи з помилками та попередженнями [6]

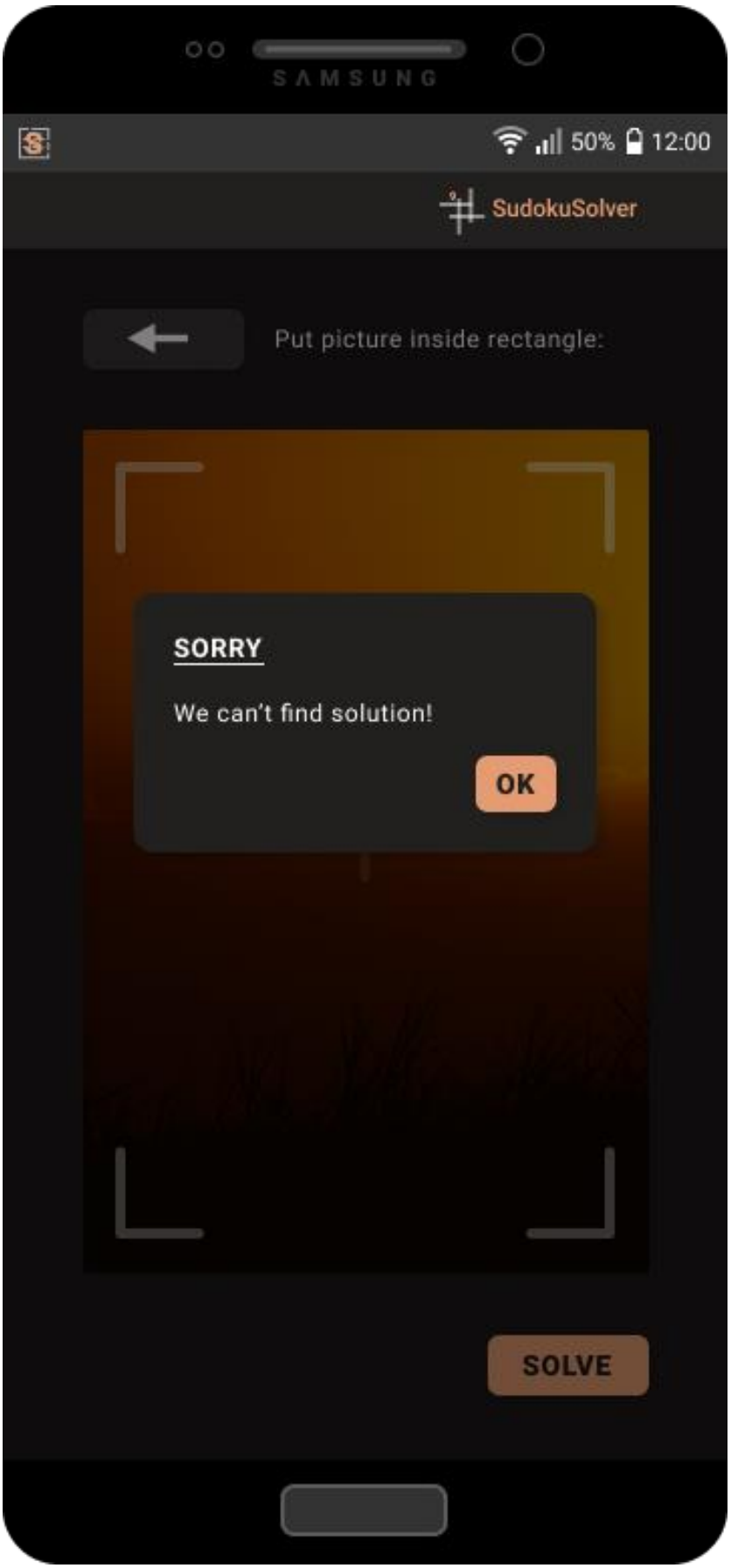

Рис 2.27. Макет відображення помилки при спробі вирішити з камери [6]

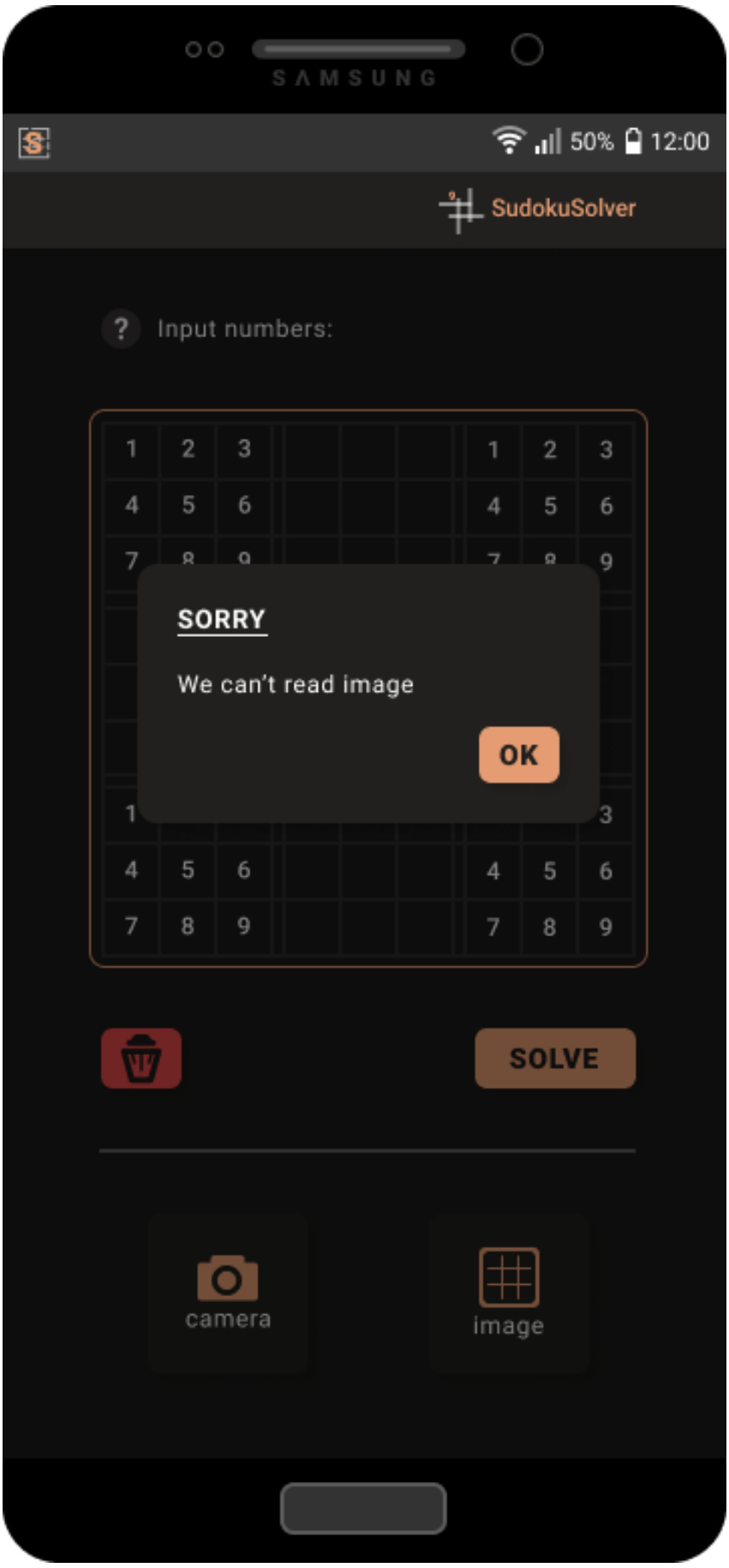

Рис 2.28. Макет відображення помилки при спробі зчитати зображення [6]

Рис 2.29. Макет відображення помилки при спробі вирішити з клавіатури [6]

Рис 2.30. Макет відображення помилки при спробі зчитати з камери [6]

Ідею та мету також було відображено на презентації дизайну (Рис 2.31).

Рис 2.31. Презентація мети та ідеї застосунку англійською мовою поряд з макетами дизайну [6]

Звучить цей текст таким чином: «Вирішальник Судоку. Класичне судоку. Цей додаток призначен для вирішення головоломки судоку за допомогою вводу користувача, зображенню з судоку або за допомогою камери. Головна ідея такого застосунку зробити вирішення пазлу більш простим та надати можливість перевіряти правильність створеного власноруч. Судоку – це дуже популярна гра, яка потребує дуже багато часу на вирішення не професійним гравцем, але з цим додатком можна вирішити його умить!»

Так чому ж такий дизайн потребує багато часу на відтворення? Все через те, що виникає необхідність створення власних елементів з дуже складною графікою. Це ставить питання про те, як оптимізувати роботу додатку та вирішити проблему адаптивного дизайну. Кожен елемент повинен мати декілька станів, анімація повинна бути плавною, а робота безперервною. Макет створювався за допомогою онлайн інструменту Figma, який допомагає лише відобразити код CSS, але мова KV потребує більше часу, щоб відобразити все по позиціям. Тому такий додаток потребує багато часу і дизайн було спрощено, а завдання на дороблення інтерфейсу користувача існує надалі.

Рис 2.32. Повний вигляд презентації у Figma [6]

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Щоб розрахувати кількість часу та ресурсів, які необхідні для розробки застосунку, було створено план розробки за днями. Кожна ітерація триває два тижня по п’ять днів на тиждень, кожен з яких триває вісім годин. На завдання подані оцінки у складності – якщо оцінка перевищує тринадцять, тоді їх необхідно розділити на більш менші.

Цей план був розрахований на одного бізнес аналітика та дизайнера або людину, яка буде відповідати за вимоги та створення мокапів; одного бекенд розробника, який відповідатиме лише за розробку функційності; одного фронтенд розробника, який відповідатиме за візуальне відображення та інтерфейс; двох інженерів з тестування, які відповідатимуть за написання будь-яких тестів та контроль якості кінцевого продукту.

Кожен представник має певні організаційні моменти, які повинні бути відображені на плані. Тому для цього було зроблено умовні позначення, які наведені нижче у Таб 3.1.

Таб 3.1. Умовні позначення процесів та їх розшифровка

CAC	create acceptance criteria
AAC	analyze acceptance criteria
ACR	acceptance criteria review
TechD	technical design
TestD	test design
BEi	backend investigation
BE	backend (functionality) development
BER	backend review
UIi	frontend investigation
UI	frontend (interface) development
UIR	frontend review
ATQCi	test investigation
QC	manual testing
AT	automation testing

Тобто усі етапи, яких необхідно дотримуватись команді, будуть відображені та кожен з них залежить від іншого, тому все повинно працювати послідовно – якщо один процес потребує якихось змін, то інший буде також зміщено відповідно до того, на скільки затримався попередній процес. Іноді деякі зміни потребують повного повертання до першого процесу і тоді, увесь процес буде перезавантажено з «нуля». Дуже важливо дотримання часових рамок, адже це впливає на уявлення замовника про професійність, а також потребує додаткових витрат з боку компанії та замовника на додатковий час розробки.

На першому тижні необхідно почати роботу над п’ятьма завданнями, та майже завершити над трьома з них (Таб 3.2). Два завдання повинні бути початі лише з боку аналізу вимог та вивчення питання з початку розробки (Таб 3.2Таб 3.3).

Таб 3.2.. Перша ітерація перший тиждень частина перша

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
Committed	**Solving Sudoku from array**	3	BA	CAC	ACR			
				AAC				
			BE	BEi	TechD		BE	
						BE		
			UI					
			ATQC	ATQCi2	TestD2		AT2	AT2
Committed	**Solving Sudoku from image**	8	BA			AAC		
					CAC	ACR		
			BE			BEi		TechD
					BEi		TechD	BE
			UI					
			ATQC			ATQCi2		
					ATQCi2			TestD2
Committed	**Add interface for solving sudoku from user input**	8	BA	CAC	ACR			
				AAC				
			BE					
			UI	UIi	UIi	TechD	UI	UI
					TechD	UI		
			ATQC		ATQCi1	TestD1	TestD1	

						ATQCi1		AT1

Таб 3.3. Перша ітерація перший тиждень частина друга

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
Committed	**Add possibility to input from image**	5	BA				CAC	ACR
							AAC	
			BE					
			UI				UIi	
							TechD	
			ATQC					TestD1
							ATQCi1	
In progress	**Add possibility to solve from camera**	5	BA					
								CAC
			BE					
			UI					
			ATQC					

На другому тижні планується закінчити роботу над трьома завданнями, які були початі на попередньому тижні та почати розробку одного з завдань і завершити його до кінця тижня (Таб 3.4, Таб 3.5). Одне завдання закінчує аналіз вимог, а два інших розпочинають аналіз вимог та вивчення питання з розробки.

Таб 3.4. Перша ітерація другий тиждень частина перша

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
Committed	**Solving Sudoku from array**	3	BA					
			BE		BER			
						BER		
			UI					
			ATQC			QC1	QC1	
Committed	**Solving Sudoku from image**	8	BA					
			BE	BE		BER	BER	
			UI					

			ATQC	TestD2 AT2	QC2	QC2	QC2	QC2

Таб 3.5. Перша ітерація другий тиждень друга перша

Committed	**Add interface for solving sudoku from user input**	8	BA					
			BE					
			UI		UIR	UIR		
			ATQC	AT1	QC1	QC1	QC1	
Committed	**Add possibility to input from image**	5	BA					
			BE					
			UI	UI	UI	UIR	UIR	
			ATQC	AT1	QC1	QC2	QC2	QC2
In progress	**Add possibility to solve from camera**	5	BA	AAC ACR				
			BE					
			UI					UIi
			ATQC					ATQCi1
Not in work	**Add possibility to change language**	3	BA		CAC	AAC ACR	ACR	
			BE					BEi TechD
			UI					
			ATQC					
Not in work	**Improve reading from image**	13	BA				CAC	CAC
			BE					
			UI					
			ATQC					

Друга ітерація передбачає активну роботу всіх робітників над п’ятьма завданнями (Таб 3.6, Таб 3.7). Ще одне завдання перейде на наступну ітерацію (Таб 3.8).

Таб 3.6. Друга ітерація перший тиждень

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
In progress	**Add possibility to solve from camera**	5	BA					
			BE					
			UI	TechD	UI	UI	UIR	
						UIR		
			ATQC	TestD1		AT1	QC1	QC1
					AT1			
Not in work	**Add possibility to change language**	3	BA					
			BE	BE		BER		
			UI					TechD
							Uli	UI
			ATQC	ATQCi2	AT2			
				TestD2	QC2			
Not in work	**Add possibility to change color schema**	3	BA			CAC	ACR	
					CAC	AAC		
			BE					
			UI					
			ATQC					ATQCi 1
Not in work	**Improve reading from image**	13	BA	ACR	ACR			
				AAC				
			BE		BEi	Tech D	BE	BE
				BEi		BE		
			UI					
			ATQC			ATQC i2	AT2	AT2
						Test D2		
Not in work	**Separate input from solution**	2	BA				CAC	AAC
								ACR

			BE					
			UI					
			ATQC					

Таб 3.7. Друга ітерація другий тиждень частина перша

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
Not in work	**Add possibility to change language**	3	BA					
			BE					
			UI	UIR				
			ATQC	QC2				
Not in work	**Add possibility to change color schema**	3	BA					
			BE					
			UI		UI			
				TechD	UIR			
			ATQC	TestD1	QC1	QC1		
Not in work	**Improve reading from image**	13	BA					
			BE	BE	BER	BER		
			UI					
			ATQC		QC2		QC1	QC2
Not in work	**Separate input from solution**	2	BA					
			BE					
			UI			UI		
						UIR		
			ATQC		TestD1		QC2	
				ATQCi1		QC1		
Not in work	**Add instruction to application**	3	BA	CAC	AAC	ACR		
					ACR			
			BE				BEi	BER
							BE	

			UI				ULi	UIR
							UI	
			ATQC			ATQCi2		QC1
						TestD2	AT2	QC2

Таб 3.8. Друга ітерація другий тиждень частина друга

Not in work	**Add possibility to solve from camera in real time**	13	BA				CAC	ACR
						CAC	AAC	
			BE					
								BEi
			UI					
								ULi
			ATQC					
								ATQCi1

Третя ітерація завершає цикл розробки програми. Останнє завдання потребує на останньому тижні лише роботу одного тестувальника, фронтенд та бекенд розробника (Таб 3.9).

Таб 3.9. Третя ітерація

Take in work	Task	SP	Part	Mo	Tu	We	Th	Fr
Not in work	**Add possibility to solve from camera in real time**	13	BA					
			BE	BEi	BE	BE	BER	
				TechD		BER		
			UI	ULi	UI	UI	UI	UIR
				TechD			UIR	
			ATQC	ATQCi1	AT1	AT1	QC1	QC1
				TestD1		QC1		

Базуючись на аналізі ринку та ситуації у країні, можна зробити визначити ризики. Для кожного ризику необхідно визначити його вірогідність та наскільки він вплине на робочий процес за десятибальною шкалою. Потім розрахувати процент небезпеки для робочого процесу перемноживши ці два значення.

Кожен передбачений ризик повинен мати заздалегідь придумане рішення, яке допоможе нормалізувати процес розробки та вкластися у зазначений бюджет. Робота над ризиками проводиться перед початком роботи та в цій дискусії приймають участь усі працівники, а менеджер є тим хто їх регулює в залежності від його оцінки ризику та неупередженості думки кожного зі співробітників.

Звіт про ризики потім презентується клієнту з метою пояснення можливих проблем під час розробки та виділення додаткового бюджету.

Таб 3.10. Ризики

Risks	Probability	Impact	Percentage	Solution
War and its repercussion can get in the way of development	10	10	100	Move all workers to safety place or hire people from peaceful places
Covid-19 still exists. Another epidemic also can come to the county	10	10	100	Allow to work remotely
Teammate get sick	5	6	30	Allow to work remotely
Unexpected private troubles	3	7	21	Relook timelines
Teammate quit	2	10	20	Have more than one worker on specialty
Technical troubles with aparature	2	5	10	Have alternative resources
Not enough resources or people	6	5	30	Have alternative resources
Wrong timelines	6	4	24	Relook timelines
Unexpected change requests	5	2	10	Relook timelines

Виходячи з інформації, яка була зпланована вище, розрахуємо бюджет, який буде запитаний у клієнта.

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}\,, \qquad (1)$$

де t - загальна трудомісткість, людино-годин; $C_{ПР}$ - середня годинна заробітна плата, грн/година.

Проаналізувавши ринок та заробітні плати [13, 14] по Україні на необхідних спеціалістів, визначимо середню заробітну плату найманих спеціалістів (Рис 3.1):

– Бізнес-аналітик: $146\,\frac{\text{грн}}{\text{година}}$.
– Фронтенд: $140\,\frac{\text{грн}}{\text{година}}$.
– Бекенд: $197\,\frac{\text{грн}}{\text{година}}$.
– Тестувальник: $117\,\frac{\text{грн}}{\text{година}}$.
– Менеджер: $58\,\frac{\text{грн}}{\text{година}}$.

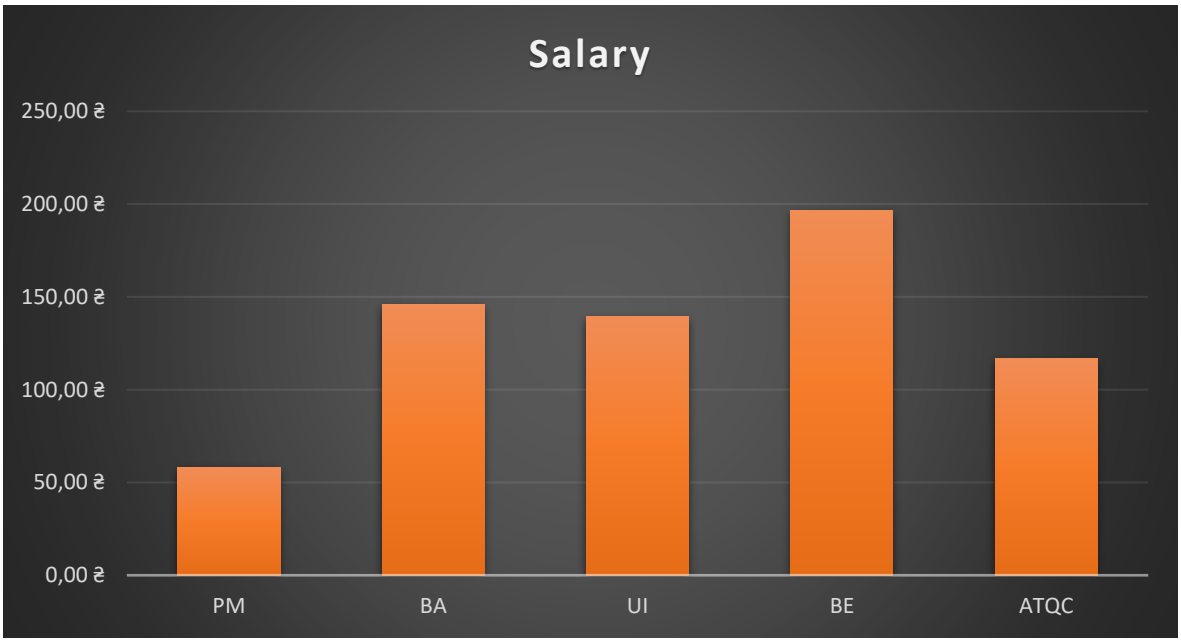

Рис 3.1. Результати аналізу ринку на предмет заробітних плат у IT-ссфері

Тому спираючись на це можна порахувати бюджет на 2,5 ітерації та закласти додатковий бюджет відповідно до ризиків. Бюджет на найманих працівників буде підраховано за формулою

$$S = \sum_{i \le n}^{i=1} З_{ЗПі}\,, \tag{2}$$

де $З_{ЗПі}$ – заробітна плата і-тої людини, а n – кількість працівників.

Повна заробітня плата за вказані строки для всіх найманих спеціалістів розрахуємо за формулою (1):

- Бізнес-аналітик: $З_{ЗП} = 160 \text{ година} \cdot \ 146 \frac{\text{грн}}{\text{година}} = 23\,360 \text{ грн.}$
- Фронтенд: $З_{ЗП} = 200 \text{ година} \cdot \ 140 \frac{\text{грн}}{\text{година}} = 28\,000 \text{ грн.}$
- Бекенд: $З_{ЗП} = 200 \text{ година} \cdot \ 197 \frac{\text{грн}}{\text{година}} = 39\,400 \text{ грн.}$
- Перший тестувальник: $З_{ЗП} = 200 \text{ година} \cdot \ 117 \frac{\text{грн}}{\text{година}} = 23\,400 \text{ грн.}$
- Другий тестувальник: $З_{ЗП} = 160 \text{ година} \cdot \ 117 \frac{\text{грн}}{\text{година}} = 18\,720 \text{ грн.}$
- Менеджер: $З_{ЗП} = 200 \text{ година} \cdot \ 58 \frac{\text{грн}}{\text{година}} = 11\,600 \text{ грн.}$

Закладений бюджет на працівників з формули (2) дорівнює:

$$S \ = 144\,480 \text{ грн}.$$

Додатково закладений бюджет ще на одну ітерацію, у разі виникнення одного з ризиків становлює 52 640 грн.

3.2. Розрахунок витрат на створення програми

Як вже було зазначено в попередньому пункті бюджет на працівників розраховано на суму 197 120 грн. Серед інших витрат є витрати на приміщення та обладнання.

Для контрольованості процесу розробки краще, щоб працівники приходили до офісів, але більшість ризиків спрямовують до того, щоб дозволити видалену роботу, якщо це можливо та необхідно.

Тому у разі, якщо буде обрана стратегія контрольованої розробки, розрахуємо усі витрати на обладнання та приміщення.

Приміщення повинно бути обладнано інтернетом, двома туалетами, місцем для харчів та місцем відпочинку. Допустимо, що приміщення не має

жодного обладнання окрім туалетів та розрахуємо вартість покупки всього необхідного для роботи у офісі команди найманих працівників (Рис 3.2).

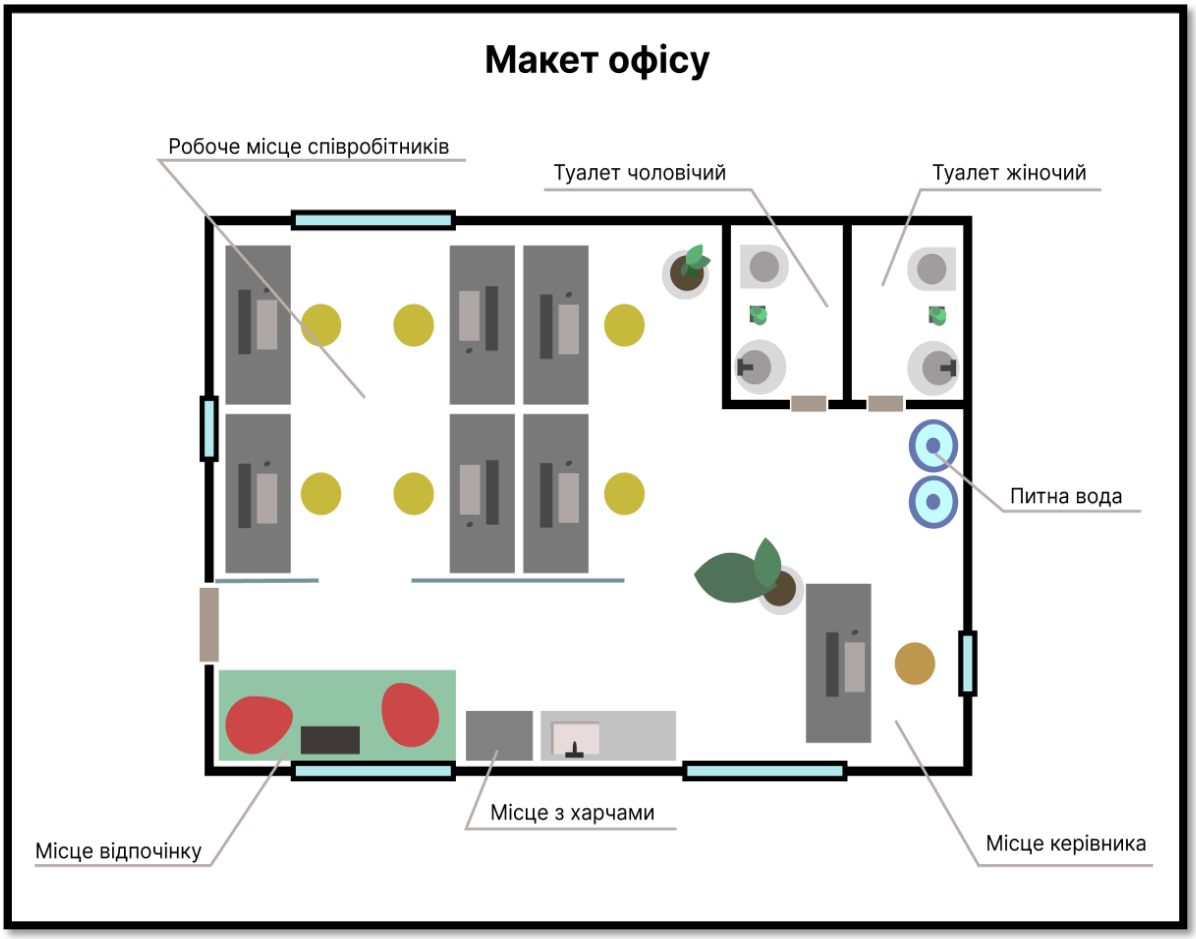

Рис 3.2. Макет офісу

Таб 3.11. Витрати

Витрати	Кількість	Одиниці	Вартість за одиницю	Всього
Оренда приміщення	133	м2	600	79800
Стіл	7	шт	897	6279
Стілець	7	шт	1999	13993
Кулер	2	шт	2550	5100
Балони з водою	60	шт по 6л	2	120
Холодильник	1	шт	5999	5999
Харчі	на 25 днів	-		7000
Полиця	2	шт	1570	3140
Шафа	1	шт	1128	1128

Пуфи	2	шт	900	1800
Рослини	4	шт	2131.5	8526
Туалетний папір	25	шт	6.875	171.875
Папір	1	шт по 500 листів	213	213
Комп'ютер (+ миша, клавіатура)	7	шт	17888	125216
Килим	1	шт	2364.1	2364.1
Всього				**260849.975**

З цього розрахунку можна зробити, що мінімальний бюджет на роботу в офісі буде становити 261 тисячу гривень за місяць. Якщо казати про роботу видалено, то виходить, що єдина витрата, яка залишається з Таб 3.11, - це комп’ютери. Тому працюючи видалено можна заощадити більше ніж у два рази.

Висновок: Очікуваний час розробки становить 200 годин, тобто 0,8 місяця. Цей термін передбачає необхідність вивчення питання розробки працівниками. З урахуванням ризиків даний період може збільшитися ще на 80 годин, тобто стати 1.06 місяця. Вартість даного програмного забезпечення 197 120 грн і вимагає лише додаткові витрати на обладнання у кількості співробітників. Тому з урахуванням обладнання бюджет становить 322 336 грн. У разі необхідності оренди приміщення, яке вже повністю обставлено, бюджет буде становити 409 641 грн. Якщо ж приміщення потребує додаткової фурнітури, повна робота на місяць буде коштувати 457 970 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи було створення мобільного застосунку, що зможе вирішувати головоломку «Судоку» з трьох видів даних – масиву цифр, файлу та камери. Робота є актуальною за технологіями, фреймворками та бібліотеками.

Призначення розробленого застосунку полягає у полегшенні вирішення головоломки новачкам та забезпеченні вирішення пазлу за допомогою комп’ютерного зору.

Під час виконання кваліфікаційної роботи було вирішено основні проблеми:

- Проаналізовано аналогічні рішення;
- На основі аналізу існуючих рішень визначено з вимогами до додатку та дизайном;
- Проаналізовано вимоги до додатку та обрано технології для швидкої та якісної реалізації;
- Спираючись на обрані технології спроектовано архітектуру додатку (технічний дизайн);
- Закладено певний час та створити план дій на розробку окремих частин проекту (окремих функцій додатку);
- Визначено та проаналізовано ризики, особливо, ті які пов’язані з необізнаністю з технологіями;
- Скорегувано у разі необхідності план дій для досягнення головної мети – створення додатку для вирішення головоломки.

Розроблена програма дозволяє:

- вирішувати головоломку «Судоку» з:
    - вводу користувача;
    - файлу зображення;
    - камери.

– Сповіщати користувача про усі помилки пов’язані з введенням та про місце помилки.

Проект реалізований на мові програмування Python з використанням фреймворку Kivy, пакетів imutils, бібліотек TensorFlow, OpenCV, Scikit-image, SciPy та py-sudoku.

Також у кваліфікаційній роботі було визначено трудомісткість розробленої системи, на базі середньої зарплати співробітників різної напрямку проведено підрахунок вартості роботи по створенню програми, який складає 197 120 грн та розраховано час на створення додатку – 200 людино-годин, тобто 0,8 місяця.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. So you thought Sudoku came from the Land of the Rising Sun ... [Електронний ресурс] / David Smith. – 2005. – Режим доступу до ресурсу: https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews.

2. Google Trends [Електронний ресурс] – Режим доступу до ресурсу: https://trends.google.com/trends/?geo=UA.

3. Sudoku solver with camera on the App Store [Електронний ресурс] – Режим доступу до ресурсу: https://apps.apple.com/us/app/sudoku-solver-with-camera/id1476185588.

4. Sudoku Solver - Solve Your Puzzles Step by Step [Електронний ресурс] – Режим доступу до ресурсу: https://www.sudoku9x9.com/smartsolver.php.

5. Best sudoku Quotes, Status, Shayari, Poetry & Thoughts | YourQuote [Електронний ресурс] – Режим доступу до ресурсу: https://www.yourquote.in/tags/sudoku/quotes.

6. Diploma.-Sudoku-Solver.-Larykova [Електронний ресурс] / Марія Ларикова. – 2022. – Режим доступу до ресурсу: https://www.figma.com/file/QEkmisOvstey9vLzKpbhbm/Diploma.-Sudoku-Solver.-Larykova?node-id=0%3A1.

7. Knuth D. Dancing Links [Електронний ресурс] / Don Knuth. – 2000. – Режим доступу до ресурсу: https://www.ocf.berkeley.edu/~jchu/publicportal/sudoku/0011047.pdf.

8. Pip [Електронний ресурс] – Режим доступу до ресурсу: https://pip.pypa.io/en/stable/.

9. Волошин А. Преимущества и недостатки языка Python [Електронний ресурс] / Алексей Волошин. – 2020. – Режим доступу до ресурсу: https://blog.ithillel.ua/ru/articles/preimushchestva-i-nedostatki-yazyka-python.

10. Kurama V. Pytorch vs. Tensorflow: Deep Learning Frameworks 2022 | Built In [Електронний ресурс] / Vihar Kurama. – 2021. – Режим доступу до ресурсу: https://builtin.com/data-science/pytorch-vs-tensorflow.

11. FAQ — Kivy 2.1.0 documentation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://kivy.org/doc/stable/faq.html#how-is-kivy-related-to-pymt.

12. Kivy vs PyQt | Guide to Top Differences of Kivy vs PyQt [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.educba.com/kivy-vs-pyqt/.

13. Середня зарплата за категорією «IT, комп'ютери, інтернет» в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://www.work.ua/salary-it/.

14. Project Manager Average Salary in Ukraine 2022 [Електронний ресурс] – Режим доступу до ресурсу: http://www.salaryexplorer.com/salary-survey.php?loc=226&loctype=1&job=326&jobtype=3.

15. MashaLar/sudokuSolver: Diploma project on Python [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/MashaLar/sudokuSolver.

ДОДАТОК А

ЛІСТИНГ

```
# main.py file

import kivy
kivy.require('1.9.1')

from kivy.app import App
from kivy.core.window import Window
from kivy.uix.screenmanager import ScreenManager, Screen
from components import SudokuWidget as SW
from camera import camera as cam
from components import SudokuGrid
from components import SudokuSquare

Window.size = (414, 700)

class SudokuApp(App):
    """Application"""
    def build(self):
        sm = ScreenManager()
        sm.add_widget(SW.SudokuScreen(name='main'))
        sm.add_widget(cam.ScreenCamera(name='camera'))
        return sm


if __name__ == '__main__':
    app = SudokuApp()
    app.run()
```

```
# sudoku.kv file
#:kivy 1.9.1
#: import sub re.sub

<SudokuScreen>:
    grid_widget: grid_widget
    canvas:
        Color:
            rgb: 0.11, 0.11, 0.11
        Rectangle:
            pos: self.pos
```

```
        size: self.size
BoxLayout:
    orientation: 'vertical'
    padding: (10,10,10,10)
    height: root.height
    width: root.width
    spacing: 5
    AnchorLayout:
        size: root.width - 20, root.height * 0.05
        anchor_x: 'right'
        anchor_y: 'top'
        size_hint_y: None
        Label:
            text: 'Sudoku puzzle:'
            text_size: self.size
            font_size: '16sp'
            halign: 'left'
            valign: 'top'
            size_hint_y: None
    SudokuGrid:
        id: grid_widget
        size: root.width - 20, root.height * 0.4
        minimum_size: [400, 200]
        size_hint_x: None
    RelativeLayout:
        size: root.width - 20, root.height * 0.4
        padding: (10,10,10,10)
        size_hint: None, None
        Button:
            halign: "left"
            text: 'CLEAN'
            color: 0,0,0,1
            bold: True
            pos: (0,  root.height * 0.325)
            size_hint: None, None
            size: 90, 30
            background_color: 0,0,0,0
            font_size: '20sp'
            text_size: self.size
            halign: 'center'
            valign: 'middle'
            on_press: root.grid_widget.clear()
            canvas.before:
                Color:
```

```
                        rgba: (1,.24,0,1) if
self.state=='normal' else (0.73,.18,0,1)  # visual
feedback of press
                    RoundedRectangle:
                        pos: self.pos
                        size: 90, 30
                        radius: [12,]
            Button:
                text: 'SOLVE'
                color: 0,0,0,1
                bold: True
                pos: (root.width - 160, root.height *
0.325)
                size_hint: None, None
                size: 130, 30
                background_color: 0,0,0,0
                font_size: '20sp'
                text_size: self.size
                halign: 'center'
                valign: 'middle'
                on_press: root.solve_button_action()
                canvas.before:
                    Color:
                        rgba: (1,.53,.39,1) if
self.state=='normal' else (0.71,.41,.32,1)  # visual
feedback of press
                    RoundedRectangle:
                        pos: self.pos
                        size: 140, 30
                        radius: [12,]
            Button:
                halign: "left"
                text: 'Export photo'
                color: 0,0,0,1
                bold: True
                pos: (50,  root.height * 0.225)
                size_hint: None, None
                size: 130, 40
                background_color: 0,0,0,0
                font_size: '18sp'
                text_size: self.size
                halign: 'center'
                valign: 'middle'
                on_release: root.export_button_action()
                canvas.before:
```

```
                    Color:
                        rgba: (1,.53,.39,1) if
self.state=='normal' else (0.71,.41,.32,1)  # visual
feedback of press
                    RoundedRectangle:
                        pos: self.pos
                        size: 130, 40
                        radius: [14,]
            Button:
                text: 'Scan sudoku'
                color: 0,0,0,1
                bold: True
                pos: (root.width - 210, root.height *
0.225)
                size_hint: None, None
                size: 130, 40
                background_color: 0,0,0,0
                font_size: '18sp'
                text_size: self.size
                halign: 'center'
                valign: 'middle'
                on_press: root.manager.current = 'camera'
                canvas.before:
                    Color:
                        rgba: (1,.53,.39,1) if
self.state=='normal' else (0.71,.41,.32,1)  # visual
feedback of press
                    RoundedRectangle:
                        pos: self.pos
                        size: 130, 40
                        radius: [14,]

            Label:
                text: 'Yes, I am mobile application,\nbut
I can have footer too!\nConfidence is when you stop
using\na PENCIL while solving SUDOKU.'
                size_hint: None, None
                pos: 70, root.height * 0.01
                halign: 'left'
                font_name: "Comic"
                font_size: '14sp'
                canvas.before:
                    Rectangle:
                        pos: -10, -40
```

```
                            size: root.width, root.height *
0.25
                            source: "./images/footer.png"

<ScreenCamera>:
    BoxLayout:
        orientation: 'vertical'
        Camera:
            id: camera
            resolution: (640, 480)
            play: False
        ToggleButton:
            text: 'Play'
            on_press: camera.play = not camera.play
            size_hint_y: None
            height: '48dp'
        Button:
            text: 'Capture'
            size_hint_y: None
            height: '48dp'
            on_press: root.capture()
        Button:
            text: 'Back'
            size_hint_y: None
            height: '48dp'
            on_press: root.manager.current = 'main'

<SudokuSquare>:
    background_color: [0.2,0.2,0.2,1]
    foreground_color: [1,1,1,1]
    selection_color:  [1,1,1,0.2]
    size: [44, 44]
    text: ''
    input_filter:
        lambda substring, from_undo: sub('\D', '',
substring) \
        [:1 - len(self.text)]
    multiline: 'False'
    font_size: '20sp'
    write_tab: False
    halign: 'center'

<LoadDialog>:
    BoxLayout:
        size: root.size
```

```
        pos: root.pos
        orientation: "vertical"
        FileChooserListView:
            id: filechooser
            path: "."
        BoxLayout:
            size_hint_y: None
            height: 30
            Button:
                text: "Cancel"
                on_release: root.cancel(root.popup)
            Button:
                text: "Load"
                on_release: root.load(filechooser.path,
filechooser.selection)

<ErrorPopup>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        Label:
            text: root.information
            text_size: self.size
            font_size: '14sp'
            halign: 'left'
            valign: 'top'
            pos: (10, 20)
            size_hint_y: 0.5
        Button:
            size_hint: None, None
            size: 80, 30
            text: "OK"
            pos: (1000, 0)
            on_release: root.ok(root.popup)
```

```
# builder.py file
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
```

```
class Model:
    @staticmethod
    def set_relu(model):
        model.add(Dense(64))
        model.add(Activation("relu"))
        model.add(Dropout(0.5))

    @staticmethod
    def set_pool(model):
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2)))

    @staticmethod
    def build(width, height, depth, classes):
        model = Sequential()

        model.add(Conv2D(32, (5, 5), padding="same",
input_shape=(height, width, depth)))
        Model.set_pool(model)
        model.add(Conv2D(32, (3, 3), padding="same"))
        Model.set_pool(model)

        model.add(Flatten())
        Model.set_relu(model)
        Model.set_relu(model)

        model.add(Dense(classes))
        model.add(Activation("softmax"))

        return model
```

```
# trainer.py file
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.datasets import mnist
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
from builder import Model
import argparse

INITIAL_LEARNING_RATE = 1e-3
EPOCHS = 10
```

```
BATCH_SIZE = 128
WH = 28 # width, height

def data_lables():
    ((dataSet, labelsSet), (data, labels)) =
mnist.load_data()

    (dataSet, data) = data_train_test(dataSet, data)

    binarizer = LabelBinarizer()
    (labelsSet, labels) = labels_train_test(labelsSet,
labels, binarizer)

    return (dataSet, labelsSet, data, labels, binarizer)

def model_setup():
    print("[INFO] CREATING MODEL...")
    optimizer = Adam(learning_rate=INITIAL_LEARNING_RATE)
    model = Model.build(width=WH, height=WH, depth=1,
classes=10)
    model.compile(loss="categorical_crossentropy",
optimizer=optimizer, metrics=["accuracy"])
    return (model)

def data_train_test(dataSet, data):
    dataSet = dataSet.reshape((dataSet.shape[0], WH, WH,
1))
    data = data.reshape((data.shape[0], WH, WH, 1))
    dataSet = dataSet.astype("float32") / 255.0
    data = data.astype("float32") / 255.0
    return (dataSet, data)

def labels_train_test(labelsSet, data, labelBinarizer):
    labelsSet = labelBinarizer.fit_transform(labelsSet)
    data = labelBinarizer.transform(data)
    return (labelsSet, data)

(dataSet, labelsSet, data, labels, binarizer) =
data_lables()
model = model_setup()

print("[INFO] LEARN ON MNIST DATASET...")
H = model.fit(dataSet, labelsSet, validation_data=(data,
labels), batch_size=BATCH_SIZE, epochs=EPOCHS, verbose=1)
```

```
print("[INFO] CHECK LEARNING...")
predictions = model.predict(data)
print(classification_report(
    labels.argmax(axis=1),
    predictions.argmax(axis=1),
    target_names=[str(n) for n in binarizer.classes_]))

print("[INFO] LOAD INTO FILE...")
model.save("./model/output/MODEL.h5", save_format="h5")
```

```
# drawer.py file
import cv2

def draw_on_image(cells, board, image):
    for (cRow, bRow) in zip(cells, board):
    for (location, number) in zip(cRow, bRow):
        x1, y1, x2, y2 = location

        textX = int((x2 - x1) * 0.33)
        textY = int((y2 - y1) * -0.2)
        textX += x1
        textY += y2

        cv2.putText(image, str(number), (textX, textY),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255),
3)
```

```
# extracter.py file
from imutils.perspective import four_point_transform
from skimage.segmentation import clear_border
import numpy as np
import imutils
import cv2

MAX_COLOR_VALUE = 255

def process_image_binary(image):
    blurred = cv2.GaussianBlur(image, (7, 7), 3)

    thresh = cv2.adaptiveThreshold(blurred,
MAX_COLOR_VALUE,
```

```
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)
    thresh = cv2.bitwise_not(thresh)
    return thresh

def get_largest_contour(image):
    thresh = process_image_binary(image)

    contours = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    contours = sorted(contours, key=cv2.contourArea,
reverse=True)
    return contours

def get_box_contour(image):
    contours = get_largest_contour(image)
    boxContour = None

    for c in contours:
        approx = cv2.approxPolyDP(c, 0.02 *
cv2.arcLength(c, True), True)
        if len(approx) == 4:
            boxContour = approx
            break

    if boxContour is None:
        raise Exception(("Can't find 4 points for
contour!"))

    return boxContour

def find_puzzle(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    puzzleContour = get_box_contour(gray)

    # apply a four point perspective transform to obtain
a top-down bird's eye view
    puzzleRGB = four_point_transform(image,
puzzleContour.reshape(4, 2))
    puzzleGray = four_point_transform(gray,
puzzleContour.reshape(4, 2))

    return (puzzleRGB, puzzleGray)
```

```
def process_image_otsu(cell):
    thresh = cv2.threshold(cell, 0, MAX_COLOR_VALUE,
        cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    thresh = clear_border(thresh)
    return thresh

def get_contour(thresh, cell):
    contours = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    if len(contours) == 0:
        return None

    mask = np.zeros(thresh.shape, dtype="uint8")
    cv2.drawContours(mask, [max(contours,
key=cv2.contourArea)], -1, MAX_COLOR_VALUE, -1)
    return mask

def percent_filled(thresh, mask):
    (h, w) = thresh.shape
    percentFilled = cv2.countNonZero(mask) / float(w * h)

    if percentFilled < 0.03:
        return None

    digit = cv2.bitwise_and(thresh, thresh, mask=mask)
    return digit

def extract_digit(cell):
    digit = None
    thresh = process_image_otsu(cell)
    mask = get_contour(thresh, cell)

    if mask is None:
        return digit

    digit =  percent_filled(thresh, mask)
    return digit
```

```
# setter.py file
from tensorflow.keras.models import load_model
```

```
from tensorflow.keras.preprocessing.image import
img_to_array
import analyzer.extracter as extr
import numpy as np
import argparse
import imutils
import cv2

def step_forward(iStep, jStep, i, j):
    x1 = i * iStep
    y1 = j * jStep
    x2 = (i + 1) * iStep
    y2 = (j + 1) * jStep
    return (x1, y1, x2, y2)

def read_board(box, modelPath):
    model = load_model(modelPath)
    puzzle = np.zeros((9, 9), dtype="int")
    iStep = box.shape[1] // 9
    jStep = box.shape[0] // 9
    cellLocations = []

    for j in range(0, 9):
        row = []
        for i in range(0, 9):
            x1, y1, x2, y2 = step_forward(iStep, jStep,
i, j)

            row.append((x1, y1, x2, y2))
            number = extr.extract_digit(box[y1:y2,
x1:x2])

            if number is not None:
                numImage = cv2.resize(number, (28, 28))
                numImage =
img_to_array(numImage.astype("float") / 255.0)
                numImage = np.expand_dims(numImage,
axis=0)

                puzzle[j, i] =
model.predict(numImage).argmax(axis=1)[0]

        cellLocations.append(row)
    return (puzzle, cellLocations)
```

```
# solver.py file
from sudoku import Sudoku
import analyzer.extracter as extr
import analyzer.drawer as dr
import analyzer.setter as st
import argparse
import imutils
import cv2

modelPath = "./model/output/MODEL.h5"
#imagePath = "./images/su.jpg"
#image = imutils.resize(cv2.imread(imagePath), width=600)
#(puzzleImage, box) = extr.find_puzzle(image)

#(board, cellLocations) = st.read_board(box, modelPath)

#print("[INFO] OCR'd Sudoku board:")
#puzzle = Sudoku(3, 3, board=board.tolist())
#puzzle.show()

#print("[INFO] solving Sudoku puzzle...")
#solution = Sudoku(3, 3, board=board.tolist()).solve()
#solution.show_full()

#dr.draw_on_image(cellLocations, solution.board,
puzzleImage)

#cv2.imshow("Sudoku Result", puzzleImage)
#cv2.waitKey(0)

def solve_on_image(image):
    (puzzleImage, box) = extr.find_puzzle(image)

    (board, cellLocations) = st.read_board(box,
modelPath)

    print("[INFO] OCR'd Sudoku board:")
    puzzle = Sudoku(3, 3, board=board.tolist())
    puzzle.show()

    print("[INFO] solving Sudoku puzzle...")
    solution = Sudoku(3, 3, board=board.tolist()).solve()
    solution.show_full()
```

```
    dr.draw_on_image(cellLocations, solution.board,
puzzleImage)
    return (solution, puzzleImage)

def get_board_from_image(imagePath):
    image = imutils.resize(cv2.imread(imagePath),
width=600)
    (puzzleImage, box) = extr.find_puzzle(image)

    (board, cellLocations) = st.read_board(box,
modelPath)

    return board.tolist()



# validator.py file
import math

def get_box_start_coordinate(x, y):
    return 3 * int(math.floor(x/3)), 3 *
int(math.floor(y/3))

def validate_sudoku(board):
    for x, i in enumerate(board):
        for y, j in enumerate(i):
            if j != 0:
                for p in range(9):
                    if p != x and j == board[p][y]:
                        return (False, f'Horizontal wrong
input in cell ({x},{y})!')
                    if p != y and j == board[x][p]:
                        return (False, f'Vertical wrong
input in cell ({x},{y})!')
                i_n, j_n = get_box_start_coordinate(x, y)
                for (i, j) in [(i, j) for p in range(i_n,
i_n + 3) for q in range(j_n, j_n + 3)
                                    if (p, q) != (x, y) and j
== board[p][q]]:
                                        return (False, f'Box
wrong input in cell ({x},{y})!')
    return (True, 'Solvable')
```

```
# camera.py file
from kivy.uix.screenmanager import ScreenManager, Screen
import time

class ScreenCamera(Screen):
    def capture(self):
        camera = self.ids['camera']
        timestr = time.strftime("%Y%m%d_%H%M%S")

camera.export_to_png("IMG_{}.png".format(timestr))
        print("Captured")

    def go_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'screen_sudoku'
```

```
# ErrorPopup.py file
from kivy.uix.floatlayout import FloatLayout
from kivy.properties import ObjectProperty
from kivy.uix.popup import Popup

class ErrorPopup(FloatLayout):
    ok = ObjectProperty(None)
    information = ObjectProperty(None)
    popup = ObjectProperty(None)

def open_error_popup(self, information):
    content = ErrorPopup(ok=self.close_popup,
information=information)
    content.popup = Popup(title="ERROR", content=content,
size_hint=(0.6, 0.2), separator_color=(1,.24,0,1),
title_size='18sp')
    content.popup.open()
```

```
# FileManager.py file
import os

from kivy.uix.floatlayout import FloatLayout
from kivy.properties import ObjectProperty
from kivy.uix.popup import Popup
```

```
from kivy.utils import platform

class LoadDialog(FloatLayout):
    load = ObjectProperty(None)
    cancel = ObjectProperty(None)
    popup = ObjectProperty(None)

def show_load(self):
    self.loadDialog = LoadDialog(load=self.export,
cancel=self.close_popup)
    PATH = "."
    if platform == "android":
      from android.permissions import
request_permissions, Permission

request_permissions([Permission.READ_EXTERNAL_STORAGE,
Permission.WRITE_EXTERNAL_STORAGE])
      app_folder =
os.path.dirname(os.path.abspath(__file__))
      PATH = "/storage/emulated/0" #app_folder
    self.loadDialog.ids.filechooser.path = PATH

    self.loadDialog.popup = Popup(title="Load file",
content=self.loadDialog,
                          size_hint=(0.9, 0.9))
    self.loadDialog.popup.open()
```

```
# SudokuSquare.py file
import kivy
kivy.require('1.9.1')

from kivy.uix.textinput import TextInput

class SudokuSquare(TextInput):
    pass


# SudokuGrid.py file
import numpy as np
import kivy
kivy.require('1.9.1')
```

```
from kivy.uix.gridlayout import GridLayout
from components import SudokuSquare as SS

class SudokuGrid(GridLayout):
    """9*9 input Grid."""

    def __init__(self, **kwargs):
        super(SudokuGrid, self).__init__(cols=3,
spacing=[2, 2], **kwargs)
        self.squares = []
        for i in range(9):
            subgrid = GridLayout(cols=3)
            for j in range(9):
                square = SS.SudokuSquare()
                subgrid.add_widget(square)
                self.squares.append(square)
            self.add_widget(subgrid)

    def to_array(self):
        user_input = [0 if square.text == '' else
int(square.text)
                      for square in self.squares]
        subgrids =
np.vsplit(np.array(user_input).reshape(27, 3), 9)
        grid = np.vstack((np.hstack(subgrids[0:3]),
                          np.hstack(subgrids[3:6]),
                          np.hstack(subgrids[6:9])))
        return grid

    def update_from_array(self, grid):
        subgrids = [np.hsplit(num, 3) for num in
np.vsplit(np.array(grid), 3)]
        output_values = np.vstack(subgrids).reshape(1,
81)[0]
        for square, value in zip(self.squares,
output_values):
            square.text = str(value)

    def update_from_image_board(self, grid):
        subgrids = [np.hsplit(num, 3) for num in
np.vsplit(np.array(grid), 3)]
        output_values = np.vstack(subgrids).reshape(1,
81)[0]
        for square, value in zip(self.squares,
output_values):
```

```
            square.text = '' if value == 0 else
str(value)

    def clear(self):
        for square in self.squares:
            square.text = ""
```

```
# SudokuWidget.py file
import kivy
import os
kivy.require('1.9.1')

from kivy.uix.widget import Widget
from kivy.properties import ObjectProperty
from sudoku import Sudoku

from components import FileManager as FileManager
from components import ErrorPopup as Error
import analyzer.validator as validator
import analyzer.solver as solver

from kivy.uix.screenmanager import ScreenManager, Screen

class SudokuScreen(Screen):
    grid_widget = ObjectProperty(None)

    def solve_button_action(self):
        self.grid = self.grid_widget.to_array()
        (solvable, information) =
validator.validate_sudoku(self.grid)
        if solvable:
            self.solve()
        else:
            Error.open_error_popup(self, information)

    def clean_button_action(self):
        self.grid_widget.clear()

    def export_button_action(self):
        FileManager.show_load(self)

    def close_popup(self, popup):
        popup.dismiss()
```

```
    def solve(self):
        puzzle = Sudoku(3, 3, board=self.grid.tolist())
        solution = puzzle.solve(raising=True)

self.grid_widget.update_from_array(solution.board)

    def export(self, path, filename):
        board =
solver.get_board_from_image(os.path.join(path,
filename[0]))
        self.grid_widget.update_from_image_board(board)

        self.close_popup(self.loadDialog.popup)
```

ДОДАТОК Б

ВІДГУК
на кваліфікаційну роботу бакалавра
на тему:
"Розробка мобільного застосунку для автоматичного розв’язування судоку на основі Python"
студентки групи 122-18-3 Ларикової Марії Володимирівни

Керівник дипломного проекту
доцент каф. ПЗКС, О.В. Реута

ДОДАТОК В

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра
на тему:
"Розробка мобільного застосунку для автоматичного розв’язування судоку на основі Python"
студентки групи 122-18-3 Ларикової Марії Володимирівни

Керівник дипломного проекту
доцент каф. ПЗКС, О.В. Реута

ДОДАТОК Г

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Ларикова_М.В.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_ Ларикова_М.В.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Таблиці_Ларикова_М.В.xml	Дані, на базі яких було створено таблиці
Програма	
Sudoku.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_ Ларикова_М.В.ppt	Презентація кваліфікаційної роботи