TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN**

**ADVANCED TOPICS IN DATA SCIENCE**

**ỨNG DỤNG SUDOKU SOLVER DEMO XỬ LÝ ẢNH**

*Người hướng dẫn*: **TS. PHẠM VĂN HUY**

*Người thực hiện*: **TRƯƠNG MINH HẬU – 51800383**

Lớp **: 18050201**

Khóa **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO MÔN**

**ADVANCED TOPICS IN DATA SCIENCE**

**ỨNG DỤNG SUDOKU SOLVER DEMO XỬ LÝ ẢNH**

*Người hướng dẫn*: **TS. PHẠM VĂN HUY**

*Người thực hiện*: **TRƯƠNG MINH HẬU – 51800383**

Lớp **: 18050201**

Khóa **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Đầu tiên, em xin cảm ơn giảng viên bộ môn lý thuyết và thực hành – thầy Phạm Văn Huy đã giảng dạy, truyền đạt những kiến thức quan trọng giúp em làm quen với các vấn đề về nghiên cứu khoa học dữ liệu và xữ lý ảnh. Nhờ lượng kiến thức của thầy truyền đạt mà đã giúp em hoàn thành thuận lợi bài báo cáo này nói riêng và tiếp thu được nhiều kiến thức quý báo.

Trong thời gian tìm hiểu và học tập bộ môn này, chắc chắn em còn rất nhiều thiếu sót và vẫn chưa tiếp thu được hết tất cả những kiến thức liên quan. Do đó, trong quá trình thực hiện bài báo cáo có những chỗ chưa được chính xác hoặc chưa đầy đủ, mong thầy bỏ qua cũng như góp ý để em hoàn thiện hơn bài báo cáo của mình.

Kính chúc thầy có nhiều sức khỏe và thành công trong công việc.

Em xin chân thành cảm ơn!

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Tp. Hồ Chí Minh, ngày tháng năm |
| (kí và ghi họ tên) |

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Tp. Hồ Chí Minh, ngày tháng năm |
| (kí và ghi họ tên) |

MỤC LỤC

[**LỜI CẢM ƠN 3**](#_Toc108695409)

[**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 4**](#_Toc108695410)

[**CHƯƠNG 1 GIỚI THIỆU TỔNG QUÁT 6**](#_Toc108695411)

[**1.1 Mô tả bài toán và ý nghĩa 6**](#_Toc108695412)

[**1.2 Các nghiên cứu liên quan 7**](#_Toc108695413)

[**CHƯƠNG 2 PHƯƠNG PHÁP, KIẾN TRÚC MÔ HÌNH, THUẬT TOÁN 9**](#_Toc108695414)

[**2.1 Naive Bayes classifier 9**](#_Toc108695415)

[**2.2 Random Forest Classifier 10**](#_Toc108695416)

[**2.3 Support Vector Machines 12**](#_Toc108695417)

[**CHƯƠNG 3 KẾT QUẢ THỰC NGHIỆM 12**](#_Toc108695418)

[**3.1 Về dữ liệu: 12**](#_Toc108695419)

[**3.2 Thực nghiệm và kết quả 13**](#_Toc108695420)

[**CHƯƠNG 4 – TỔNG KẾT 23**](#_Toc108695421)

[**TÀI LIỆU THAM KHẢO 24**](#_Toc108695422)

# ỨNG DỤNG DEMO – SODUKU SOLVER

## Mô tả ứng dụng

Đây là một ứng dụng demo đơn giản để áp dụng các vấn đề xử lý ảnh vào trong thực tế.

Ứng dụng gồm 2 phần Backend và Frontend:

* Backend sử dụng ngôn ngữ Python để giải quyết các vấn đề về xử lý ảnh như đọc ảnh, cắt ảnh, chuyển ảnh thành chữ số, … với các thư viện chủ yếu được sử dụng như: opencv (thư viện xử lý ảnh), easyocr (nhận dạng hình ảnh quang học thành chữ viết), flask (thư viện giúp phát triển code Python ở BE theo dạng API để giao tiếp với giao diện FE).
* Frontend đơn giản được trên nền tảng web với các ngôn ngữ đơn thuần như HTML, css, javascript, … có nhiệm vụ gửi hình ảnh về cho BE xử lý, sau đó nhận lại giá trị các ô số của sudoku mà BE trả về qua các API và hiển thị lên trên web.

## Chi tiết ứng dụng

Về giao diện ứng dụng:

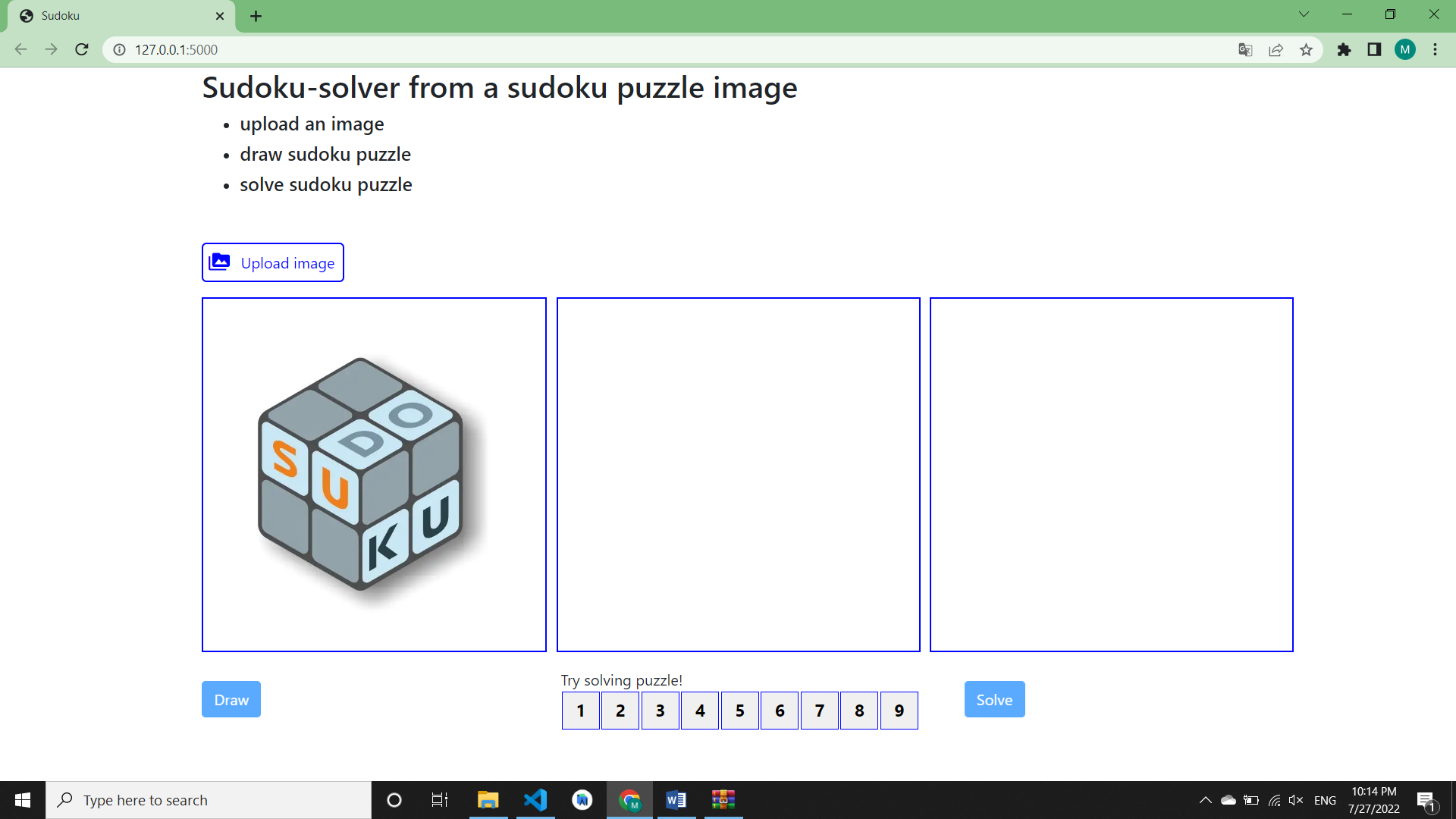


Figure 1 – Giao diện ứng dụng

Gồm các chức năng:

* Upload ảnh gửi về Backend xử lý.
* Từ dữ liệu trả về vẽ ra sudoku và cho phép người dùng tự giải như một trò chơi điện tử.
* Solve sudoku đã vẽ.

Các hình ảnh upload lên là hình ảnh các sudoku như sau (hiện tại chỉ nhận ảnh chỉ chứa sudoku, chưa phát triển tính năng nhận dạng sudoku trên một hình ảnh phức tạp):

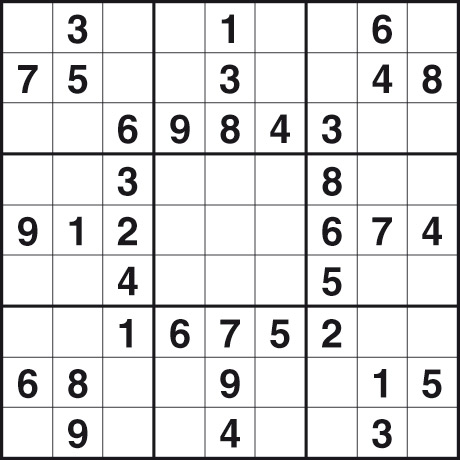


Figure 2 – Hình ảnh sudoku test

Click vào nút “Upload image” để chọn ảnh

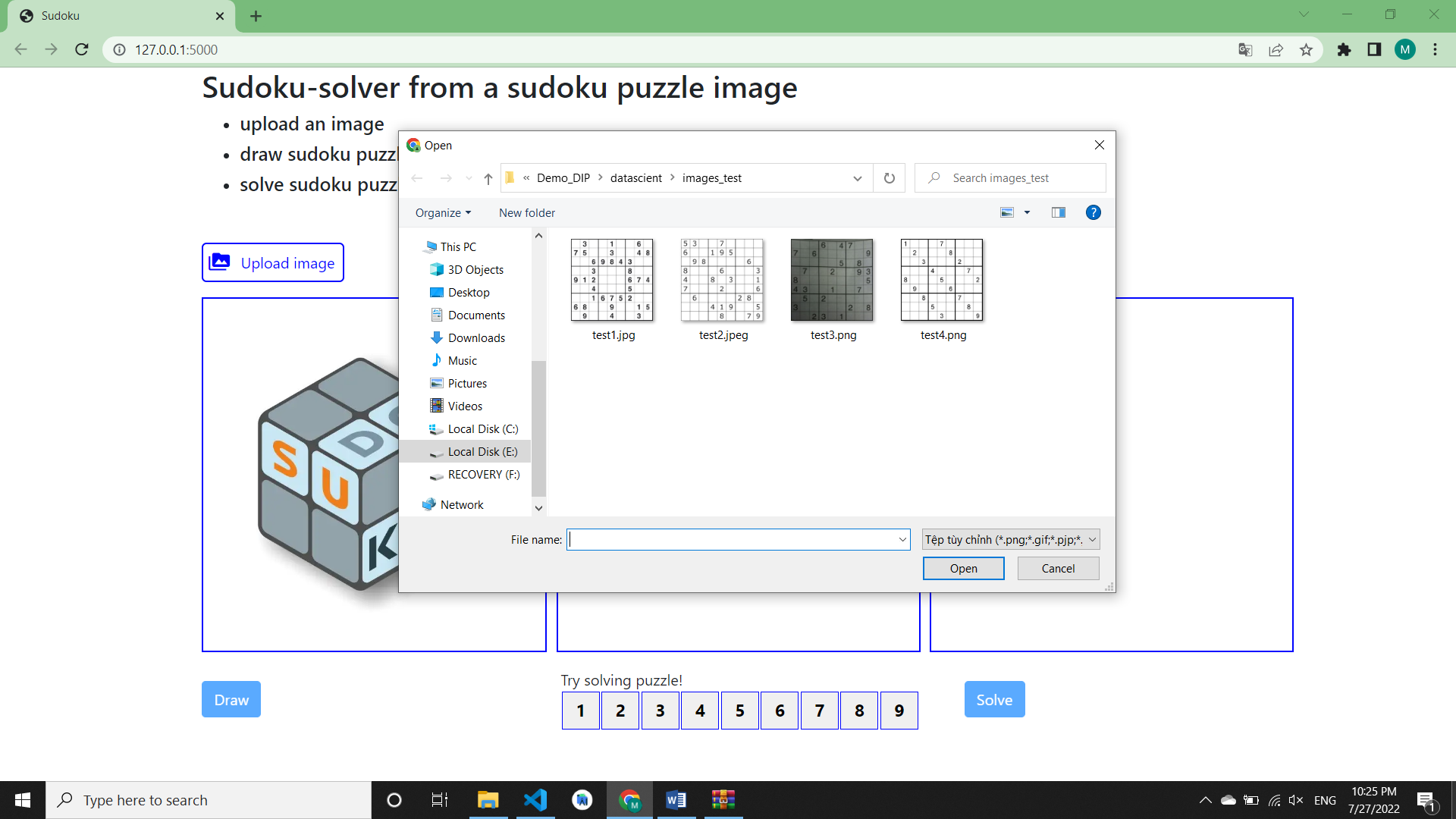


Figure 3 – Chọn ảnh

Sau khi chọn ảnh sẽ hiển thị như sau:

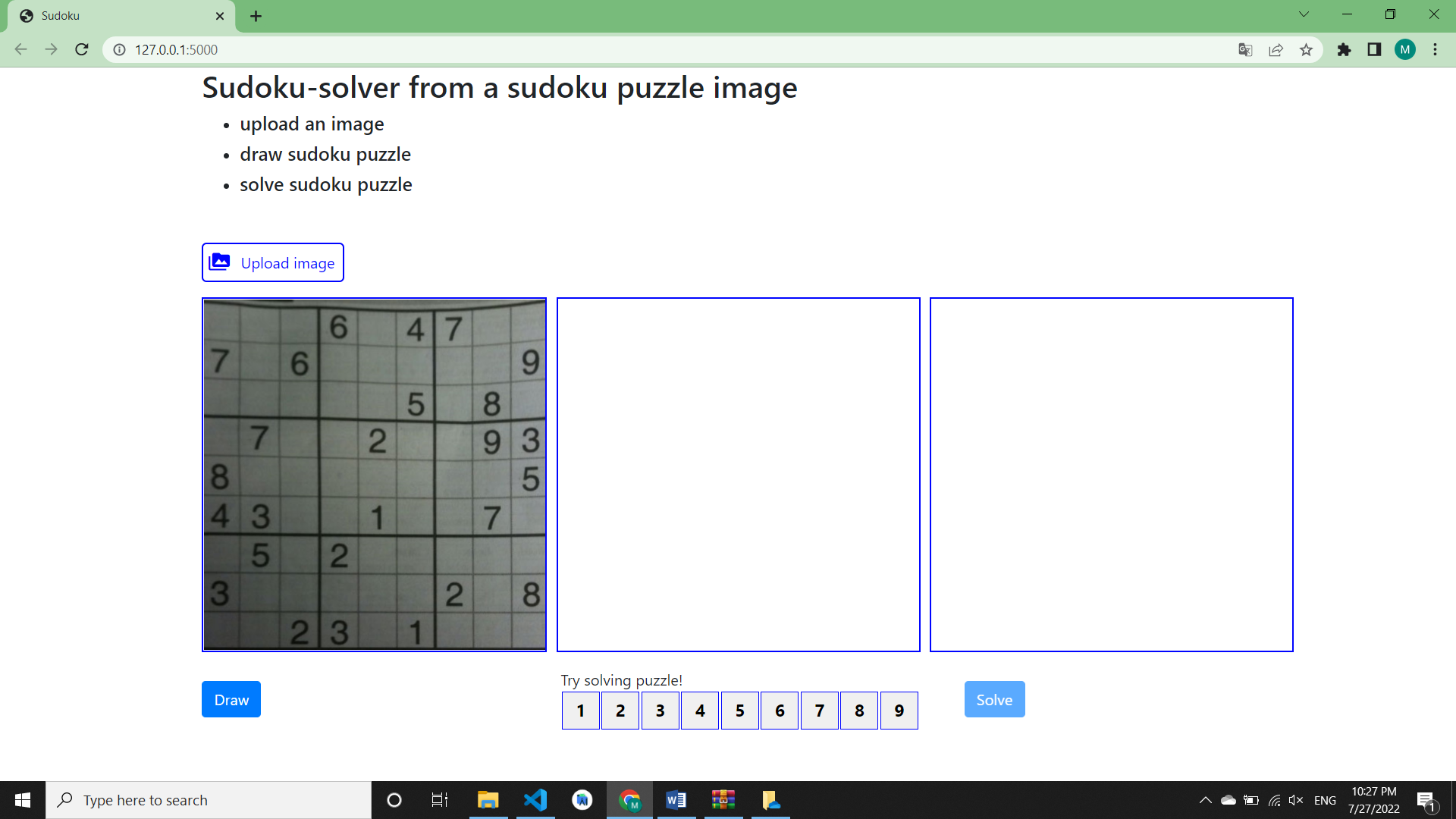


Figure 4 – Hiển thị ảnh

Click vào “Draw” để gửi ảnh về Backend xử lý và lấy kết quả để vẽ ra sudoku

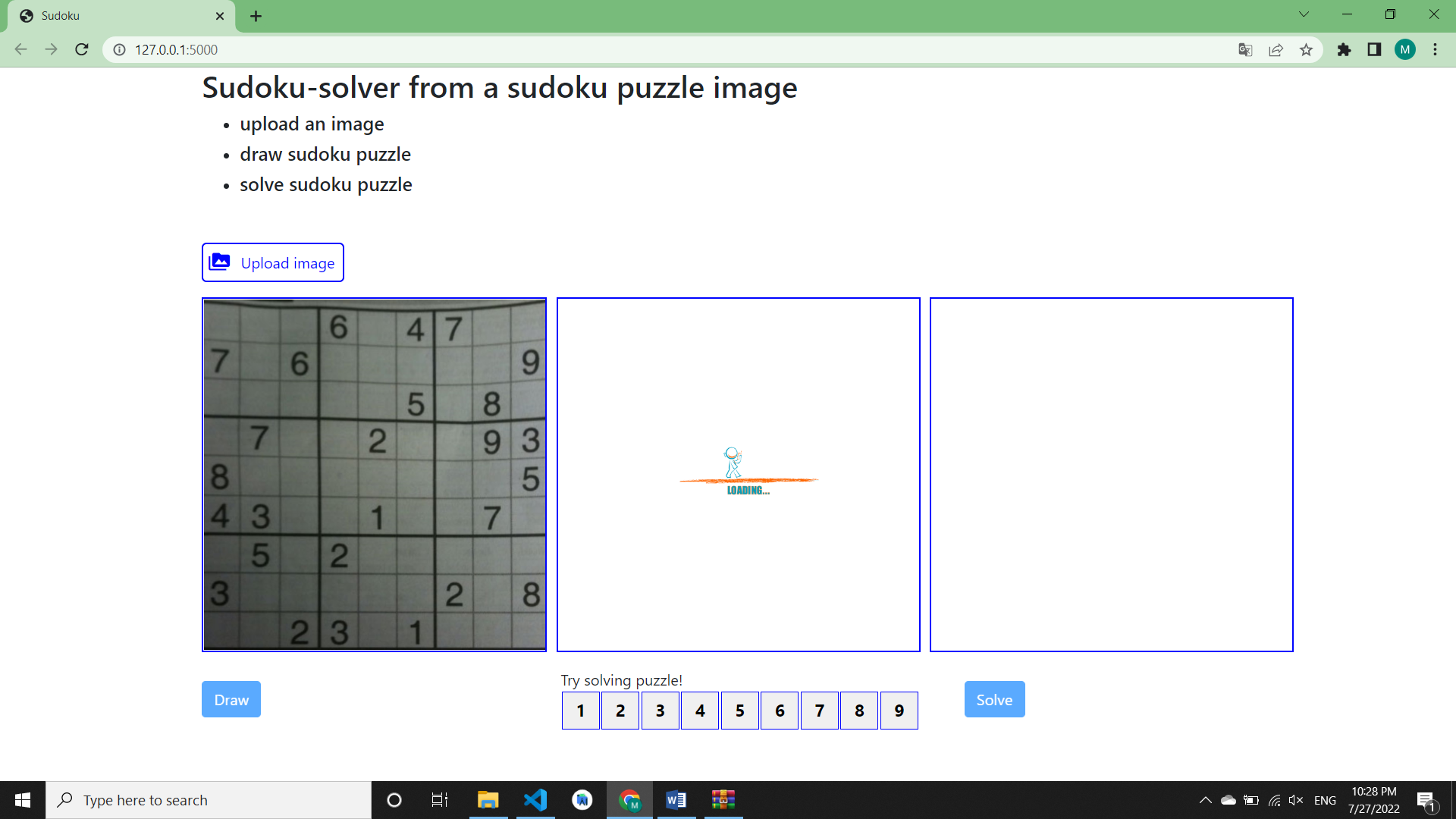


Figure 5 – Loading ảnh

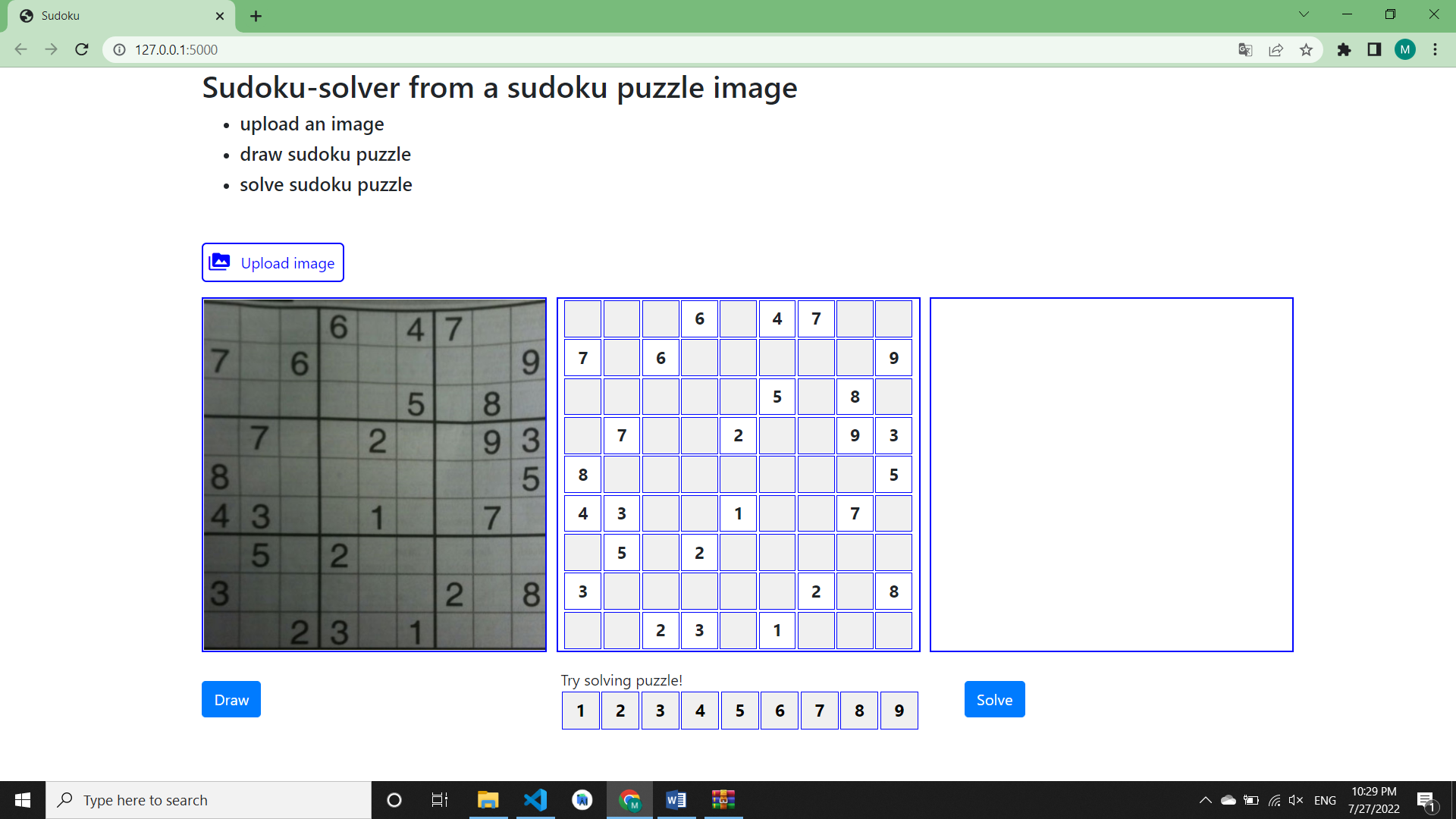


Figure 6 – Vẽ sudoku từ kết quả BE trả về

Ở đây chúng ta có thể thực hiện việc trải nghiệm giải sudoku bằng cách chọn các ô trống và chọn các số từ 1 tới 9 ở phía dưới để điền vào.

Tiếp theo để xem kết quả solve sudoku nhấn vào “Solve”

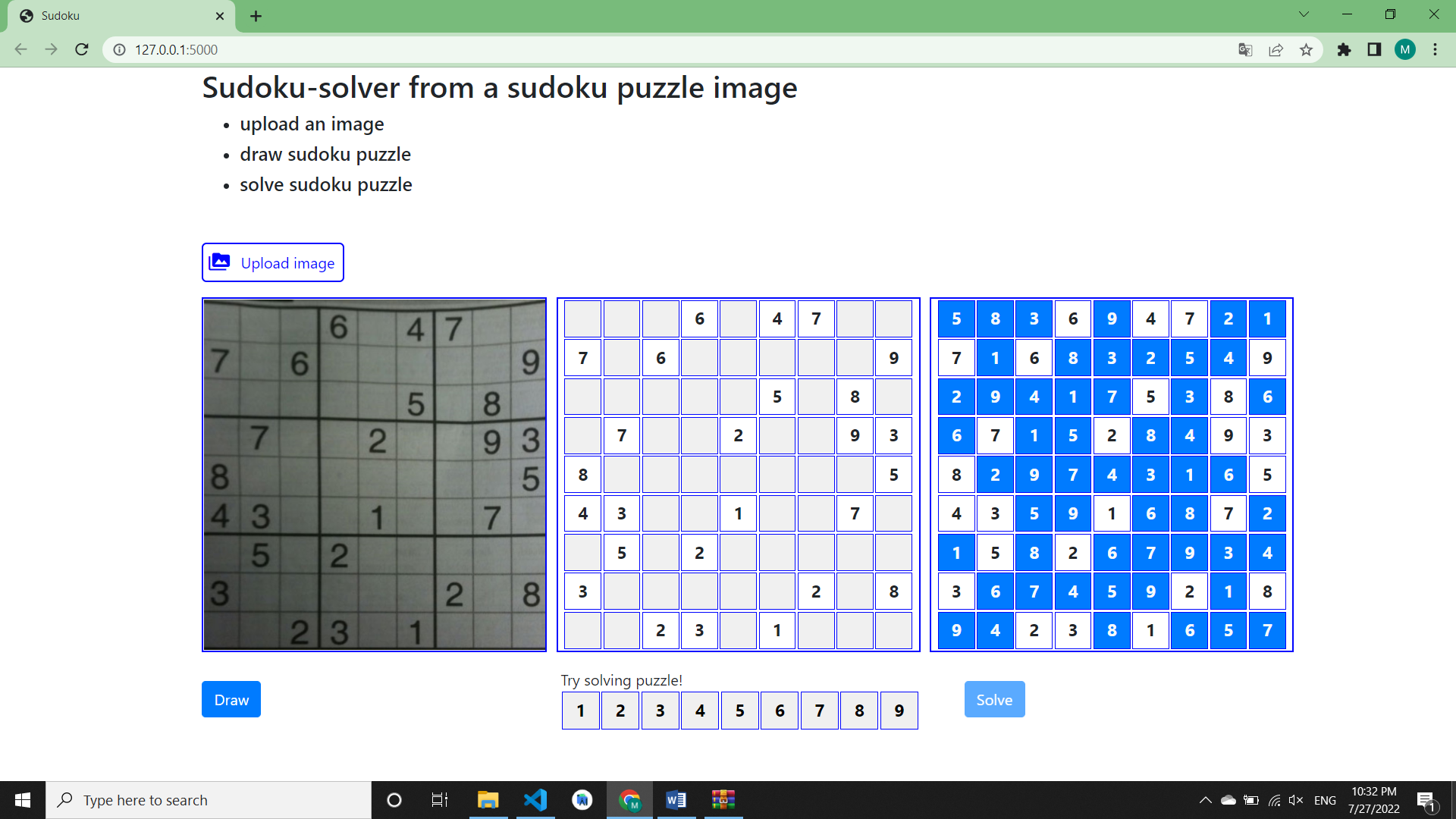


Figure 7 – Kết quả solve sudoku

Phía Backend gồm các API như:

* POST : /upload

Nhận ảnh từ FE gửi về lưu vào thư mục uploads.

@app.route('/upload', methods=['POST'])

def upload():

    x = datetime.datetime.now()

    dateTimeStr = str(x).replace('-', '').replace(':',

                                                  '').replace(' ', '').replace('.', '')

    file = request.files['file']

    filename = dateTimeStr+'\_'+file.filename

    file.save(os.path.join('upload/'+filename))

    return filename

* GET: /get\_all\_cells

Ở đây từ ảnh vừa upload sẽ gọi hàm read\_img() và return\_cells() để trả về chuỗi các số trong sudoku.

@app.route("/get\_all\_cells", methods=['GET'])

def get\_cells():

    image = request.args.get('image')

    uri = 'upload/'+image

    read\_img(uri)

    sleep(1)

    gird\_not\_solve = return\_cells()

    global global\_gird

    global\_gird = gird\_not\_solve

    return json.dumps({"gird\_not\_solve": gird\_not\_solve})

Hàm read\_img() đọc ảnh đầu vào và thực hiện các biến đổi xử lý, cắt nhỏ ảnh từng ô và lưu lần lượt tất cả các ô vào thư mục BoardCells với tên từng ảnh là cell<số hàng><số cột> (ví dụ cell00.jpg)

def read\_img(image):

    # read image

    image\_url = image

    img = cv2.imread(image\_url)

    # crop cells

    grid\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  # =>gray image

    grid = cv2.bitwise\_not(cv2.adaptiveThreshold(

        grid\_gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 57, 5))  # =>thresh image

    edge\_h = np.shape(grid)[0]

    edge\_w = np.shape(grid)[1]

    celledge\_h = edge\_h // 9

    celledge\_w = np.shape(grid)[1] // 9

    global w, h

    w = celledge\_w

    h = celledge\_h

    tempgrid = []

    for i in range(celledge\_h, edge\_h + 1, celledge\_h):

        for j in range(celledge\_w, edge\_w + 1, celledge\_w):

            rows = grid[i - celledge\_h:i]

            tempgrid.append([rows[k][j - celledge\_w:j]

                            for k in range(len(rows))])

    #

    finalgrid = []

    for i in range(0, len(tempgrid) - 8, 9):

        finalgrid.append(tempgrid[i:i + 9])

    #

    for i in range(9):

        for j in range(9):

            finalgrid[i][j] = np.array(finalgrid[i][j])

    try:

        for i in range(9):

            for j in range(9):

                os.remove("BoardCells/cell" + str(i) + str(j) + ".jpg")

    except:

        pass

    for i in range(9):

        for j in range(9):

            cv2.imwrite(str("BoardCells/cell" + str(i) +

                        str(j) + ".jpg"), finalgrid[i][j])

Hàm return\_cells() là hàm trả về giá trị là một mảng 2 chiều chứa các giá trị của các ô sudoku. Giá trị các ô được lấy bằng các nhận dạng quang học các hình ảnh của từng ô đã xử lý và lưu trong thư mục BoardCells thông qua thư viện easyocr.

def return\_cells():

    rows = []

    all = []

    reader = Reader(['en'])

    k = 0

    for i in range(9):

        for j in range(9):

            img = cv2.imread(str("BoardCells/cell" + str(i) +

                                 str(j) + ".jpg"))

            kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (4, 2))

            img2 = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)

            img3 = cv2.blur(img2, (3, 3))

            img3 = img3[0:h, 10:w]  # scale follow number

            # read number from image

            num = reader.readtext(img3, allowlist='0123456789')

            if(num == []):

                print(0, end=' ')

                rows.append(0)

            else:

                print(num[0][1], end=' ')

                rows.append(int(num[0][1]))

            k = k+1

            if(k == 9):

                all.append(rows)

                rows = []

                print()

                k = 0

    return all

Hình ảnh các cell khi đã sử lý và cắt nhỏ:

E:\_TRUONGMINHHAU\DataScience\Demo_DIP\datascient\BoardCells\cell40.jpg

Figure 8 – cell40.jpg

* GET: /solve

Trả về kết quả sudoku đã giải

@app.route("/solve", methods=['GET'])

def get\_solve():

    gird = global\_gird.copy()

    if (Sudoku(gird, 0, 0)):

        print\_sudoku(gird)

    else:

        print("Can not solve!")

    return json.dumps({"gird": gird})

Gồm các hàm để giải sudoku:

def solve(grid, row, col, num):

    for x in range(9):

        if grid[row][x] == num:

            return False

    for x in range(9):

        if grid[x][col] == num:

            return False

    startRow = row - row % 3

    startCol = col - col % 3

    for i in range(3):

        for j in range(3):

            if grid[i + startRow][j + startCol] == num:

                return False

    return True

def Sudoku(grid, row, col):

    if (row == M - 1 and col == M):

        return True

    if col == M:

        row += 1

        col = 0

    if grid[row][col] > 0:

        return Sudoku(grid, row, col + 1)

    for num in range(1, M + 1, 1):

        if solve(grid, row, col, num):

            grid[row][col] = num

            if Sudoku(grid, row, col + 1):

                return True

        grid[row][col] = 0

    return False

Cấu trúc thư mục source:

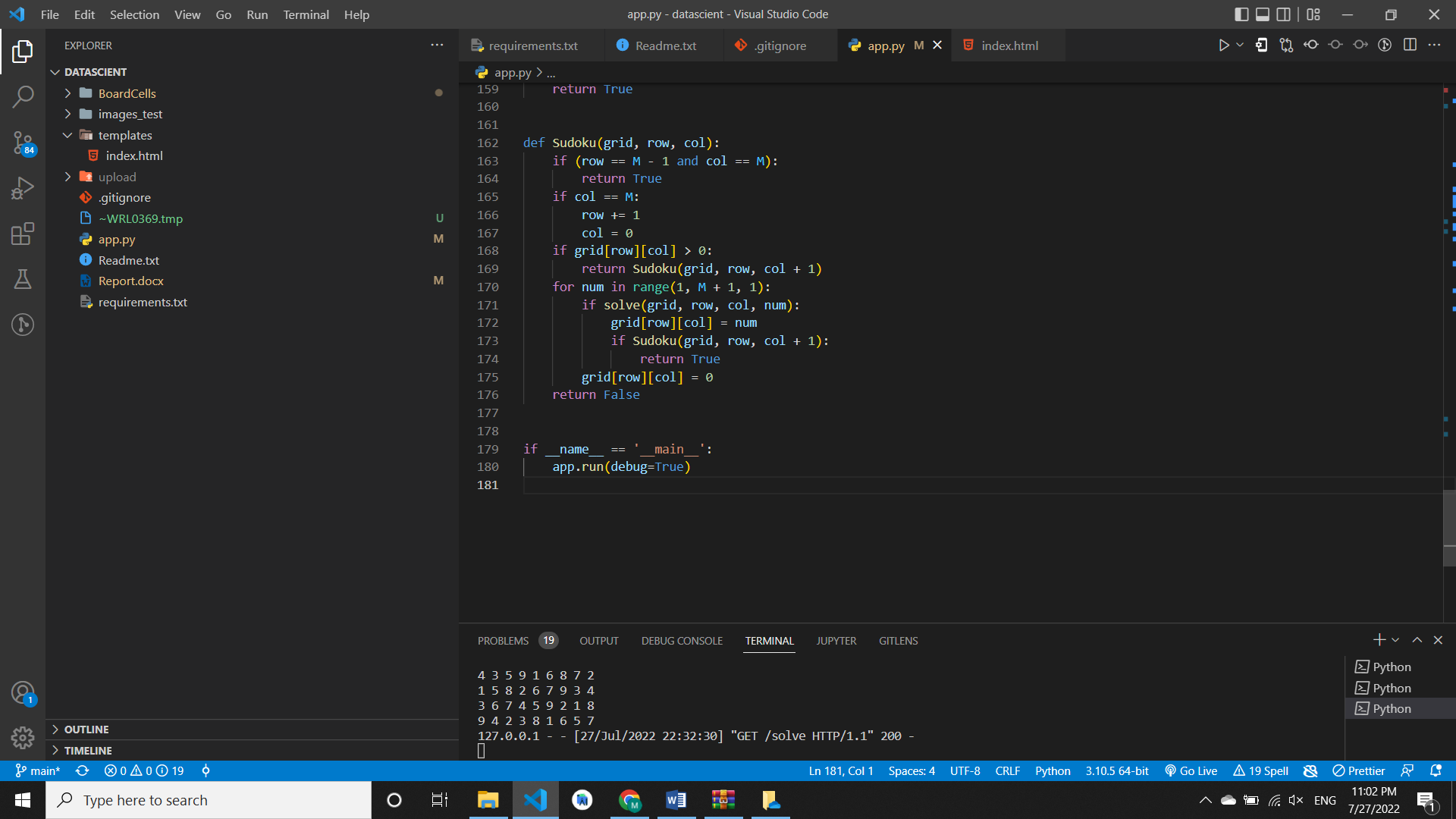


Figure 9 – Thư mục source

Gồm các phần chính:

* Thư mục upload chứa ảnh được upload lên từ FE
* Thư mục images\_test chứa các ảnh có thể dùng để test ứng dụng
* Thư mục BoardCells chứa các hình ảnh từng ô sau khi xử lý và cắt ra.
* Thư mục templates chứa file index.html là code giao diện
* File app.py là code Python – BE xử lý app.

Cách chạy ứng dụng :

* Máy đảm bảo đầy đủ các thư viện sử dụng trong app
* Chạy file app.py để chạy server
* Vào địa chỉ localhost để sử dụng web (địa chỉ thường là : http://127.0.0.1:5000)

# TỔNG KẾT

Qua bài báo cáo đã trình bày về ứng dụng của xử lý ảnh bằng ngôn ngữ Python trong một bài toán thực tế.

Ứng dụng demo tuy đơn giản nhưng phần nào đó cho thấy sức mạnh của xử lý ảnh, nhận dạng chữ số, …

Ứng dụng có kết hợp các thư viện xử lý ảnh, các công nghệ cấu tạo thành một trang web đơn giản.

# TÀI LIỆU THAM KHẢO

1. <https://medium.com/becoming-human/image-processing-sudokuai-opencv-45380715a629>
2. <https://www.javatpoint.com/flask-tutorial>