BROAD EXPLANATION OF HOW EVERYTHING WORKS TOGETHER :
FRONTEND (HTML, CSS, JavaScript) vs BACKEND : Python

Interaction user/sudoku => JavaScript functions handle the selection of cells

example : selectCell function

So at the Javascript level : the function will remove the previous selection and add the selected CSS class to the clicked cell which changes the background color of the cell
. selected class

another function for incorrect : example .wrong class

Then, the HTML/CSS level : visually represent the selected and incorrect cells => clouds

The Flask Python level :
If someone enters a number in a cell and click a button to check the sol => JavaScript sends an AJAX request to the Flask backend

The flask backend receives the request and processes the Sudoku grid data
Once puzzle solved, validated => backend sends a response containing info about incorrect cells back to the frontend along with the solved puzzle if applicable

Finally, on the integration : AJAX callback function in the frontend receives the response from the backend which includes data about the solved puzzle and incorrect cells.

By integrating frontend Javascript code with backend Flask code through AKAX requests => interface dynamic and responsive

SO :

frontend Javascript : user interaction + visual representation
Backend flask code : processes data, performs logic such as solving puzzle, identifying cells, sends responses …

1. Highlighting cells with CSS and Javascript

At the html and the CSS level we will need to use functions of this form :

```
.selected {
   background-color: lightblue;
}

.wrong {
   background-color: #FFCCCC;
}
```

At the Javascript level : we need to write javascript functions allowing us to handle the selection of cells in the sudoku grid

FOR SELECTION :

```
function selectCell(row, col) {
   // Remove previous selection
   $('.selected').removeClass('selected');

   // Highlight the selected cell
   $('table tr:eq(' + row + ') td:eq(' + col + ')').addClass('selected');
}
```

FOR INCORRECT :

```
// Function to highlight incorrect cells
function highlightIncorrectCells(incorrectCells) {
   for (var i = 0; i < incorrectCells.length; i++) {
      var cell = incorrectCells[i];
      $('table tr:eq(' + cell.row + ') td:eq(' + cell.col + ')').addClass('wrong');
   }
}
```

At the Python level (FLASK),

we will need a data validation => validate user input and determine incorrect answers
AND to send data to frontend : send info about incorrect cells from the backend to the frontend

```
@app.route('/solve', methods=['POST'])
def solve():
   # Get Sudoku grid data from request
   sudoku_grid = request.json['grid']

   # Solve Sudoku puzzle and get incorrect cells
   solved_grid, incorrect_cells = solve_sudoku(sudoku_grid)

   # Return solved puzzle and incorrect cells to frontend
   return jsonify({'solved_grid': solved_grid, 'incorrect_cells': incorrect_cells})
```

At the AJAX requests level : we need to use AJAX to send data from frontend to backend + receive responses containing info about incorrect cells

```
// AJAX request to solve Sudoku puzzle and highlight incorrect cells
$.ajax({
   type: 'POST',
   url: '/solve',
   data: JSON.stringify({ grid: sudokuGrid }),
   contentType: 'application/json',
   success: function(response) {
      // Handle response from backend
      var solvedGrid = response.solved_grid;
      var incorrectCells = response.incorrect_cells;
      updateGridDisplay(solvedGrid);
      highlightIncorrectCells(incorrectCells);
   },
   error: function(xhr, status, error) {
      console.error(error);
   }
});
```

Research user database

database to store user info => usernames, passwords …

SQL databases like MySQL, PostgreSQL, SQLite …

either own database server either database-as-a-service provider like Firebase Authentication, Amazon Cognito, or Auth0 => authentication services along with user databases

THEN : backend implementation

/register : registering new users
/login
/logout

Frontend :
User registration and login : forms => user input and send to backend endpoints
Javascript for form validation and to handle form submissions => AJAX requests

FOR USER DATABASE :

either we build our own : SQL or NoSQL
either Database as a service providers
either third party authentication : Github for example

HTML Templates

AND NEXT TO THIS : HTML TEMPLATES :

examples : welcome.html, login.html, main.html => all inside the templates folder.

HTML PAGES WOULD LOOK LIKE THIS :

welcome.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Welcome</title>

<head>
<body>
        <h1>Play Sudoku!</h1>
        <a href=‘’/login”>Login</a>

</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Login</title>

<head>
<body>
        <h1>Login</h1>

</body>
</html>
```

main.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Main Page</title>
```

```
<head>
<body>
        <h1>Now it’s your turn!</h1>
        <!— sudoku grid, buttons, timer …>

</body>
</html>
```

Implementation of the keypad

Creating HTML elements for keypad buttons and writing JavaScript functions to handle user interactions.

HTML structure :

on click => calls a Javascript function to handle button click event

```
<div id="keypad"> <button onclick="selectNumber(1)">1</button> <button
onclick="selectNumber(2)">2</button> <button onclick="selectNumber(3)">3</button> <button
onclick="selectNumber(4)">4</button> <button onclick="selectNumber(5)">5</button> <button
onclick="selectNumber(6)">6</button> <button onclick="selectNumber(7)">7</button> <button
onclick="selectNumber(8)">8</button> <button onclick="selectNumber(9)">9</button> <button
onclick="clearCell()">Clear</button> </div>
```

JavaScript fonctions :

Handle selecting numbers from keypad / clearing currently selected cell

```
// Function to handle selecting a number from the keypad
function selectNumber(number) {
   // Get the currently selected cell
   var selectedCell = $('.selected');

   // Check if a cell is selected
   if (selectedCell.length > 0) {
      // Update the value of the selected cell with the selected number
      selectedCell.text(number);
   }
}

// Function to handle clearing the currently selected cell
function clearCell() {
   // Get the currently selected cell
   var selectedCell = $('.selected');

   // Check if a cell is selected
   if (selectedCell.length > 0) {
      // Clear the value of the selected cell
      selectedCell.text('');
   }
}
```

CSS Styling : for the appearance of the keypad

```
#keypad {
   display: flex;
   flex-wrap: wrap;
}
```

```css
#keypad button {
   width: 30px;
   height: 30px;
   margin: 5px;
   font-size: 16px;
}
```

Integration :

selectNumber() clearCell() functions

-Interactive interface web : difficult : design, put interactivity …
- -wireframes / user interface mockups
- -frontend: HTML, CSS? Javascript frameworks
- -interactive features : input validation, highlighting selected cells, feedback if incorrect

-hint system : quite hard
- -integrate hint to solver algo
- -user interface for requesting / displaying hints

Homepage :
-welcoming message / introduction or description of the game
-navigation links to different sections : « Play Sudoku », « About », « Instructions », « Contact »

Play Sudoku page :
Sudoku grid => 9x9 sudoku grid with blank cells / clear borders separating the 3x3 sub grids
Try to highlight the currently selected cell
=> provide a keypad ? or only with keyboard

Options for selecting difficulty levels
Button for checking the solution, solving the puzzle, restart it, generating a new puzzle
=> 4 buttons : check solution, solve puzzle, restart, new puzzle

Timer to keep track of how long it takes to solve it

Hint button !
Highlight incorrect answers in red

Why not something to save the progress and resume the puzzle => especially if log-in

About page :
-Introduce rules of sudoku
-Background info on dev of website

Instructions page :
-instructions on how to play Sudoku for beginners
-Tips / strategies for solving

Contact page :
-Feedback / support
-Mail for inquiries

Settings Page :
-Allow users to customise the appearance of the sudoku grid
-switch between different themes

FLASK :

pip install Flask

from flask import Flask, render_template, request

```
app = Flask(__name__)

@app.route('/')
def index():
        return render_template('index.html')

@app.route('/solve', methods=['POST'])
def solve():
        return ''sudoku solved"

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

create HTML templates for the different pages : as much as interfaces
=> folder 'templates' in the project directory and HTML files inside
example : index.html for the page

HOW WOULD WE CREATE DIFFERENT PAGES IN THE APP?
Define multiple routes, each corresponding to a different page.

=> modify app.py to include routes for welcome page, main page, login page …

example :

```
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

@app.route('/')
def welcome():
    return render_template('welcome.html')

@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/main')
def main():
    return render_template('main.html')

@app.route('/solve', methods=['POST'])
def solve():
        return « Sudoku solved »
if __name__ == '__main__':
    app.run(debug=True)
```

AND NEXT TO THIS : HTML TEMPLATES :

examples : welcome.html, login.html, main.html => all inside the templates folder.

HTML PAGES WOULD LOOK LIKE THIS :

welcome.html

```
<!DOCTYPE html>
<html>
<head>
	<title>Welcome</title>

<head>
<body>
	<h1>Play Sudoku!</h1>
	<a href=‘’/login”>Login</a>

</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html>
<head>
	<title>Login</title>

<head>
<body>
	<h1>Login</h1>

</body>
</html>
```

main.html

```
<!DOCTYPE html>
<html>
<head>
	<title>Main Page</title>

<head>
<body>
	<h1>Now it’s your turn!</h1>
	<!— sudoku grid, buttons, timer …>

</body>
</html>
```

THEN

We could add links and buttons in the templates to go from one page to there other
example : link welcome.html to go to login page, and link in login page to go to main page

FOR THE AESTHETIC ASPECT :

We can use JavaScript => clicking solve buttons, putting numbers into the Sudoku grid

in JavaScript => AJAX request to a specific endpoint in the Flask app
=> would contain the data representing the current state of the sudoku grid.
Would receive data, solve sudoku puzzle, send back solved puzzle.

CSS to do aesthtetic choies for the web app interface

Example for AJAX with jQuery => send Flask backend :
On javascript :

```
// Assume this function is called when the user clicks a "Solve" button
function solveSudoku() {
    // Get the current state of the Sudoku grid (for example, as a 2D array)
    var sudokuGrid = getGridData();

    // Make an AJAX request to your Flask backend
    $.ajax({
        type: "POST",
        url: "/solve",
        data: JSON.stringify({ grid: sudokuGrid }), // Send the Sudoku grid data
        contentType: "application/json",
        success: function(response) {
            // Handle the response from the backend (e.g., display the solved puzzle)
            console.log(response);
        },
        error: function(xhr, status, error) {
            // Handle any errors
            console.error(error);
        }
    });
}
```

=> /solve route
afte solving puzzle => send back the solved puzzle or any relevant info as the response.

= way to make the sudoku interactive

FOR THE KEYPAD :

one possibility is creating a set of buttons with each number from 1 to 9 + a clear button to remove a number from a cell.
=> each button : JavaScript function to update corresponding cell in sudoku grid.

html :

```
<!DOCTYPE html>
<html>
<head>
    <title>Main Page</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script type="text/javascript">
        // Function to handle selecting a number from the keypad
        function selectNumber(number) {
            // Update the currently selected cell in the Sudoku grid with the selected number
            // You'll need to implement this part
        }

        // Function to handle clearing the currently selected cell
        function clearCell() {
            // Clear the currently selected cell in the Sudoku grid
            // You'll need to implement this part
        }
    </script>
</head>
<body>
    <h1>Sudoku Solver</h1>

    <!-- Sudoku grid display -->
    <!-- Keypad -->
    <div id="keypad">
        <button onclick="selectNumber(1)">1</button>
        <button onclick="selectNumber(2)">2</button>
        <button onclick="selectNumber(3)">3</button>
        <button onclick="selectNumber(4)">4</button>
        <button onclick="selectNumber(5)">5</button>
        <button onclick="selectNumber(6)">6</button>
        <button onclick="selectNumber(7)">7</button>
        <button onclick="selectNumber(8)">8</button>
        <button onclick="selectNumber(9)">9</button>
        <button onclick="clearCell()">Clear</button>
    </div>
</body>
</html>
```

in the previous code : each button in the keypad calls the selectNumber function with the corresponding number when clicked
selectNumber function : updates currently selected cell in the sudoku grid with selected number
clearCell : responsible for clearing currently selected cell in sudoku grid

FOR THE BUTTONS :

solving, resetting, checking solution
=> styles with CSS

FOR THE TIMER :
JavaScript
When user starts solving the puzzle => timer begins and stops when puzzle solved.
Javascript code to update the timer display at regular intervals

HINT BUTTON :
helps to solve the puzzle
JavaScript code can make an AJAX request to Flask backend : hint for current state of puzzle
Backend would then compute a hint and send it back to the frontend

JavaScript code would then update the Sudoku grid display with the hint

html :

```
<!DOCTYPE html>
<html>
<head>
    <title>Main Page</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script type="text/javascript">
        // Function to handle solving Sudoku puzzle
        function solveSudoku() {
            // AJAX request to solve the puzzle
            $.ajax({
                type: "POST",
                url: "/solve",
                data: JSON.stringify({ grid: sudokuGrid }),
                contentType: "application/json",
                success: function(response) {
                    // Handle the solved puzzle (update grid display, etc.)
                    console.log(response);
                },
                error: function(xhr, status, error) {
                    console.error(error);
                }
            });
        }

        // Function to handle requesting a hint
        function requestHint() {
            // AJAX request to get a hint
            $.ajax({
                type: "GET",
                url: "/hint",
                success: function(response) {
                    // Handle the hint (update grid display, etc.)
                    console.log(response);
                },
                error: function(xhr, status, error) {
                    console.error(error);
                }
            });
        }

        // Function to update the timer display
        function updateTimer() {
            // Update timer display
            // You'll need to implement this part
        }

        // Start the timer when the page is loaded
        $(document).ready(function() {
            setInterval(updateTimer, 1000); // Update timer every second
        });
    </script>
</head>
<body>
    <h1>Sudoku Solver</h1>

    <!-- Sudoku grid display -->
    <!-- Buttons for actions like solving, resetting, etc. -->
    <!-- Hint button -->
    <button onclick="solveSudoku()">Solve</button>
    <button onclick="requestHint()">Hint</button>
    <!-- Timer display -->
    <div id="timer">00:00:00</div>
</body>
</html>
```

In Flask backend : define a new route : ‘/hint’
Would compute hint for current state of sudoku and send it back to frontend

On python :

```
@app.route(‘/hint’)
def hint()
        # TO IMPLEMENT
        hint = compute_hint(current_state_puzzle)
        return jsonify(hint)
```

HIGHLIGHTING CURRENT CELL AND WRONG ANSWER CELL

=> CSS + modification of html ad javascript

html :

```
<!DOCTYPE html>
<html>
<head>
    <title>Main Page</title>
    <style>
        /* Style for the Sudoku grid */
        table {
            border-collapse: collapse;
        }
        td {
            border: 1px solid black;
            width: 30px;
            height: 30px;
            text-align: center;
            vertical-align: middle;
            cursor: pointer;
        }
        .selected {
            background-color: lightblue; /* Style for the selected cell */
        }
        .wrong {
            background-color: #FFCCCC; /* Style for incorrect answers */
        }
    </style>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script type="text/javascript">
        // Function to handle selecting a cell in the Sudoku grid
        function selectCell(row, col) {
            // Remove previous selection
            $('.selected').removeClass('selected');

            // Highlight the selected cell
            $('table tr:eq(' + row + ') td:eq(' + col + ')').addClass('selected');
        }

        // Function to handle selecting a number from the keypad
        function selectNumber(number) {
            // Get the selected cell
            var selectedCell = $('.selected');

            // Check if a cell is selected
            if (selectedCell.length > 0) {
                // Update the value of the selected cell
                selectedCell.text(number);

                // Remove wrong class if present
                selectedCell.removeClass('wrong');
            }
        }

        // Function to handle clearing the currently selected cell
        function clearCell() {
            // Get the selected cell
            var selectedCell = $('.selected');

            // Check if a cell is selected
            if (selectedCell.length > 0) {
                // Clear the value of the selected cell
                selectedCell.text('');
            // Remove wrong class if present
            selectedCell.removeClass('wrong');
        }
    }
  </script>
</head>
<body>
  <h1>Sudoku Solver</h1>

  <!-- Sudoku grid display -->
  <table>
    <tr>
      <td onclick="selectCell(0, 0)"> </td>
      <!-- Include all cells in the Sudoku grid with appropriate onclick attribute -->
      <!-- For example: <td onclick="selectCell(0, 1)"> </td> -->
    </tr>
    <!-- Include all rows in the Sudoku grid -->
  </table>

  <!-- Keypad -->
  <div id="keypad">
    <button onclick="selectNumber(1)">1</button>
    <!-- Include all number buttons in the keypad -->
    <!-- For example: <button onclick="selectNumber(2)">2</button> -->
    <button onclick="clearCell()">Clear</button>
  </div>
</body>
</html>
```

CSS classes : .selected and .wrong
selectCell => highlighting selected cell by adding it to the .selected class

selectNumber : updates value of selected cell in the sudoku grd when number button clicked
clearCell : clears value of selected cell when clear button clicked

incorrect answers : .wrong

BASED ON THE NEW YORK TIME SUDOKU WE WOULD LIKE :

A FIRST PAGE:

-Presentation
-Choice of level of difficulty
-Log-in

A SECOND PAGE :

-Sudoku interface
-Keypad
-Timer
-Highlight of the cell we are on
-highlight of the cell in red if wrong

-a resume option if we leave the website
=> local storage to store and retrieve state of Sudoku Puzzle

+4917671253969
+491744710135

Stories :

1. Make wireframes or mockups
2. Frontend of the application with HTML, CSS and JavaScript
3. Hint system with the solver algorithm / adaptating the interface for these hints
4. Designing the homepage and the different navigation links
5. Play sudoku page with highlights, keypad, buttons …
6. About page with rules and background info
7. Flask Integration with different routes for the different pages
8. Html templates for the different pages inside templates folder of the project directory
9. CSS Sytling for the HTML templates (aesthetic)
10. AJAX : send datea to the Flask backend for slving, receiving hints, updating the sudoku…
11. Implementing the keypad (from 1 to 9 + clear button)
12. Button functionality
13. Timer implementation with JavaScript to update the timer at regular interval
14. Hint button implementation (JavaScript code, AJAX)
15. Highlighting cells with CSS and Javascript
16. Local storage integration to store and take the state of the sudoku puzzle (resume option)

SUDOKU RULES :

STEPS ARE :

-sudoku grid : number from 1 to 9
-but this grid must be valid => algorithm : we could fill it and remove numbers => backtracking
-need a function to display the sudoku grid
-need a function to check if a solution is valid and respect the rules
-check if sudoku puzzle solved correctly

each row contains nb 1 to 9
each column contains nb 1 to 9
each 3x3 subgrid contains the nb 1 to 9

Starting with empty Sudoky grid

=> <u>backtracking algorithm</u>

iterative traversal: looking at each cell of sudoku grid one by one from the top left corner to the bottom right corner
ensures that we examine very cell in the grid to determine what number should be palced there

recursive placement: which number to place there while making sure it follows sudoku rules
try nb from 1 to 9. If a nb fits the rules we palce it in the cell
Backtrack to previous cell and try diff number if doesn't work

backtracking : taking a step back if stuck
If we reach a dead end where no nb can be placed in a cell without breaking sudoku rules, backtrack to previous cell where we made a choice and try a different number

Fill each cell with a random nb satisfying sudoku rules

= iteration over each cell
number from 1 to 9

If grilled grid violates any rule => try another number until valid grid obtained

Add randomization : to have different sudoku puzzles, shuffle order in which numbers are tried at each cell

[How to generate Sudoku boards with unique solutions]

1. Start with an empty board.
2. Add a random number at one of the free cells (the cell is chosen randomly, and the number is chosen randomly from the list of numbers valid for this cell according to the SuDoKu rules).
3. Use the backtracking solver to check if the current board has at least one valid solution. If not, undo step 2 and repeat with another number and cell. Note that this step might produce full valid boards on its own, but those are in no way random.
4. Repeat until the board is completely filled with numbers.

1. Start with a complete, valid board (filled with 81 numbers).
2. Make a list of all 81 cell positions and shuffle it randomly.
3. As long as the list is not empty, take the next position from the list and remove the number from the related cell.
4. Test uniqueness using a fast backtracking solver. My solver is - in theory - able to count all solutions, but for testing uniqueness, it will stop immediately when it finds more than one solution.
5. If the current board has still just one solution, goto step 3) and repeat.
6. If the current board has more than one solution, undo the last removal (step 3), and continue step 3 with the next position from the list
7. Stop when you have tested all 81 positions.

https://stackoverflow.com/questions/6924216/how-to-generate-sudoku-boards-with-unique-solutions

8. Start with a complete, valid board (filled with 81 numbers).
9. Make a list of all 81 cell positions and shuffle it randomly.
10. As long as the list is not empty, take the next position from the list and remove the number from the related cell
11. Test uniqueness using a fast backtracking solver. My solver is - in theory - able to count all solutions, but for testing uniqueness, it will stop immediately when it finds more than one solution.
12. If the current board has still just one solution, goto step 3) and repeat.
13. If the current board has more than one solution, undo the last removal (step 3), and continue step 3 with the next position from the list
14. Stop when you have tested all 81 positions.

Make exercise more interesting :

We could create different sudoku with difficulty levels => adjusting the pre-filled cells in the initial grid

We could also do an interesting/interactive interface by using a graphical library

We could add a hint possibility

https://www.101computing.net/sudoku-generator-algorithm/

Should have unique solution

Fill a 9x9 grid with digits so that each column, each row and each of the nine 3x3 sub grids that compose the grid contain all of the digits from 1 to 9

=> how to adapt algo to estimate difficulty level of a sudoku grid
=> should different algo be used to generate sudoku grids for a specific for a specific difficulty level ?

1. generate a full grid of numbers fully filled in : so different than the other websit
More complex as it seems as we cannot just randomly generate numbers to fill in the greed. Must respect the rules. Backtracking algorithm that we will apply to an empty grid.
2. from full grid, will have to remove 1 value at a time
3. each time a value is removed we will apply a sudoku solver algorithm to see if the grid can still be solved and to count the nb of sol it leads to.
4. if resulting grid has one sol we can carry on the process from step 2. if not, put value we took away back in the grid.
5. can repeat same process from step 2 several times using a different value each time to try to remove additional numbers, resulting in a more difficult grid to solve. Number of attempts we will use to go through this process will have an impact on the difficulty level of the resulting grid.

https://gamedev.stackexchange.com/questions/56149/how-can-i-generate-sudoku-puzzles
Take a puzzle generator application to have different patterns, with different difficulty …
Generate a seed puzzle

Then randomising the numbers ? Rotate to have more ? Flop horizontally, vertically, or both
=> have different variations

///

Backtracking algorithms

General algorithmic technique that considers searching every possible combination in order to solve a computational problem.

=> build a solution incrementally, one piece at a time, removing those sol that fail to satisfy the constraints of the problem at any point of time.

=> A perfect example is SudoKo
filling digits one by one => current digit cannot lead to a sol => remove it and try next digit