by Ali Tarek

Aly Tarek Portfolio.

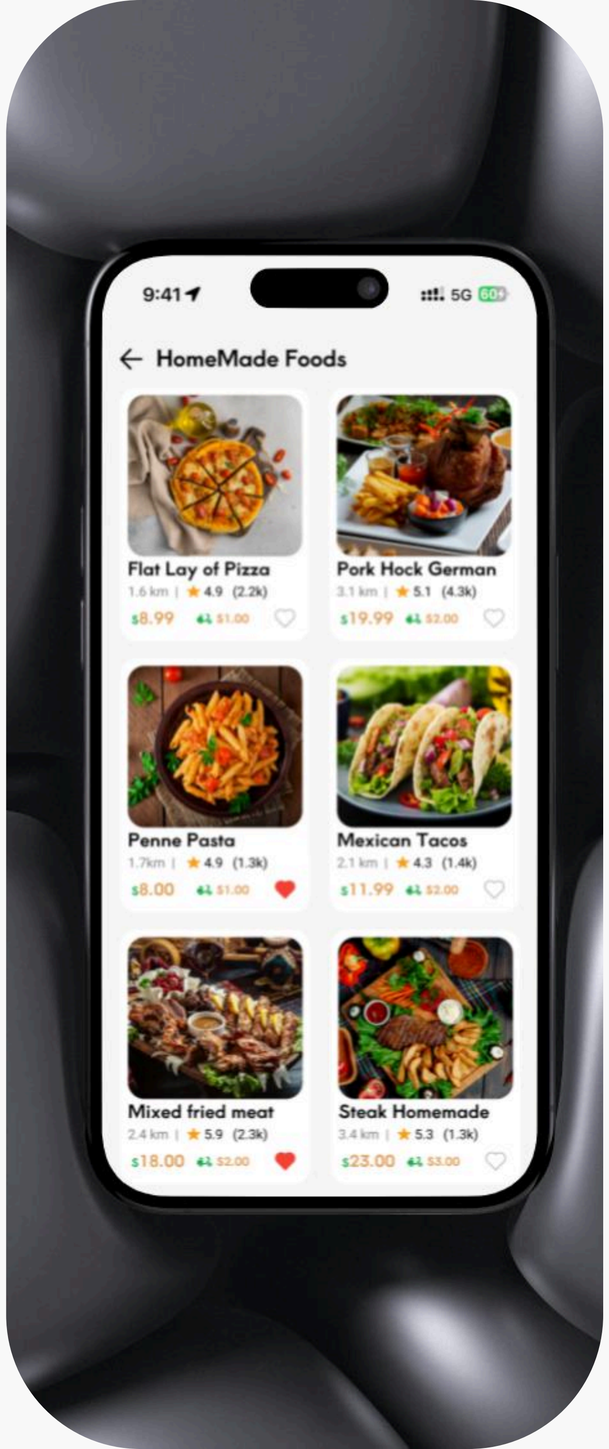
Table of-Contents.

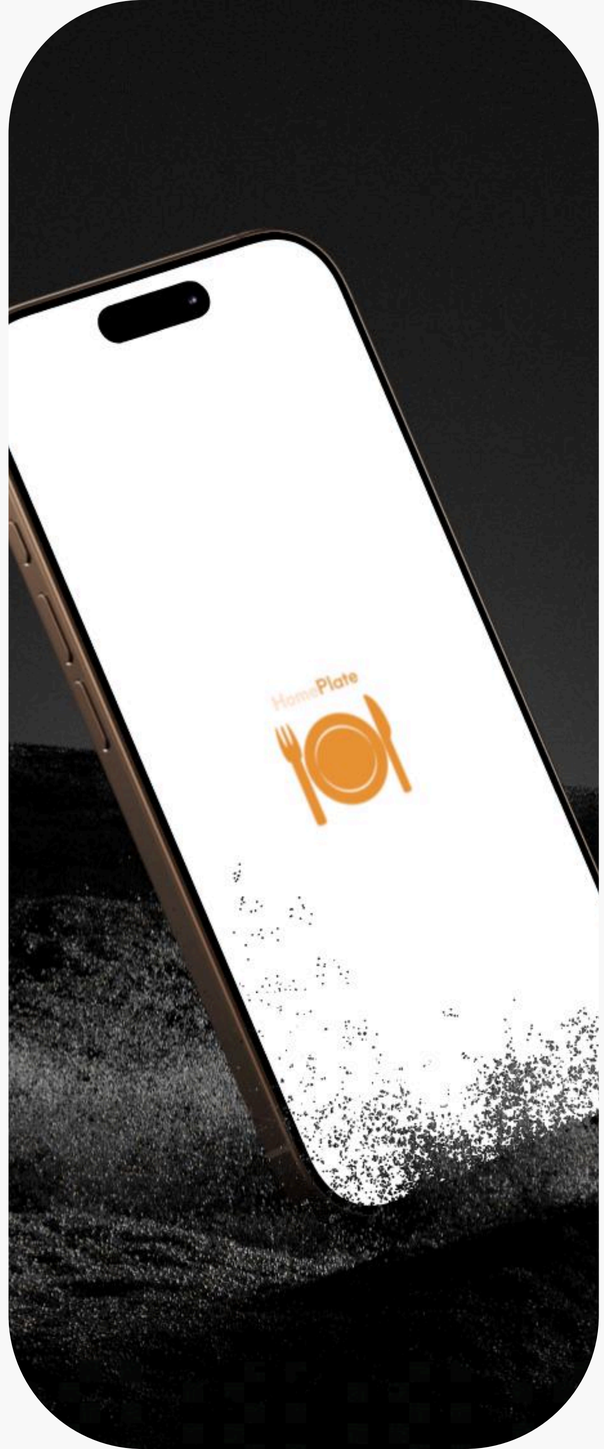
1. HomePlate App Design
2. HomePlate App Code ShowCase
3. Queuer App Design
4. Queuer App Code ShowCase
5. Python Programs
6. Get In Touch
7. Thank You

Project

HomePlate Design

For this project, the goal was to create a fresh, modern Mobile Application design identity for a new tech startup. It began by conducting research to understand the target audience and the competitive landscape. From there, we developed a design strategy and visual identity that reflected the company's innovative spirit and forward-thinking approach. The end result was a cohesive brand identity that effectively communicates the company's mission and values.

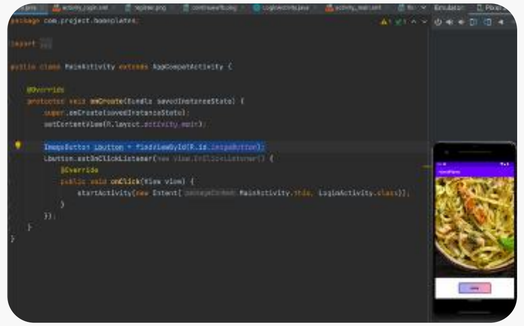


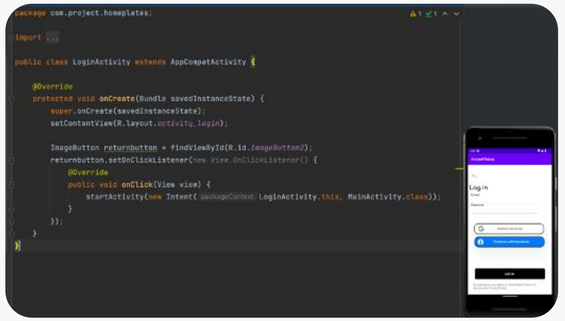


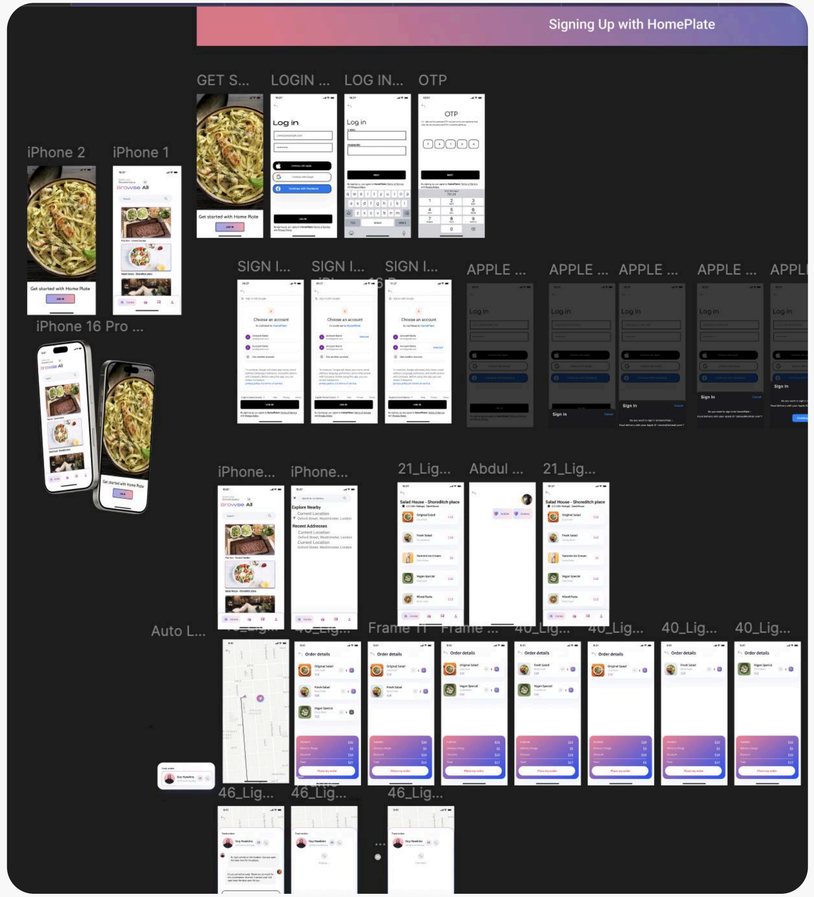
Project

HomePlate Java Code

Figure 1







The MainActivity class is the entry point of an Android project, defined in the com.project.application package. It imports AppCompatActivity, a base class for activities with support library features, and Bundle, a container for passing data between Android components. By extending AppCompatActivity, MainActivity inherits useful methods. The onCreate() method is overridden to initialize the activity, calling super.onCreate(savedInstanceState) to ensure proper setup and setContentView(R.layout.activity_main) to link the activity to its XML layout file, setting up the first screen of the app.

Project

→

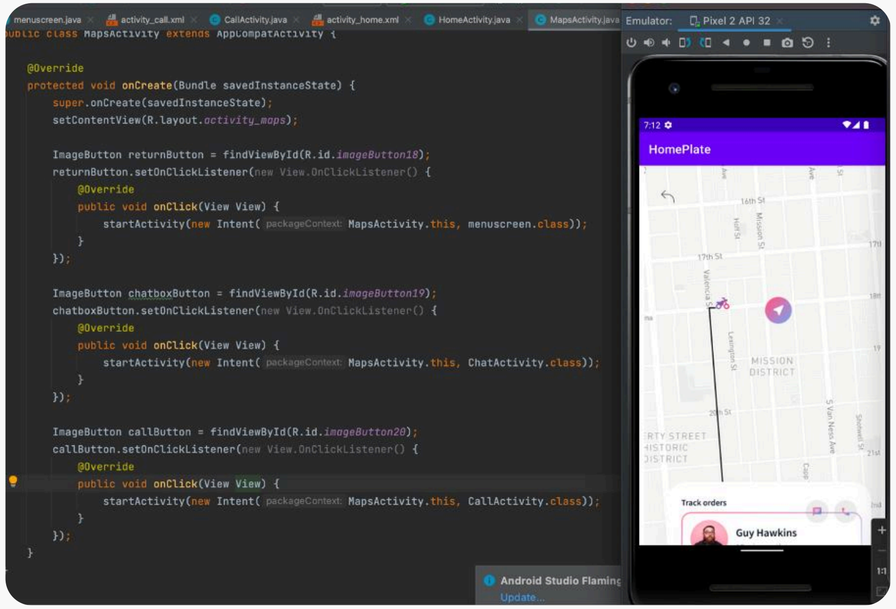
HomePlate Java Code

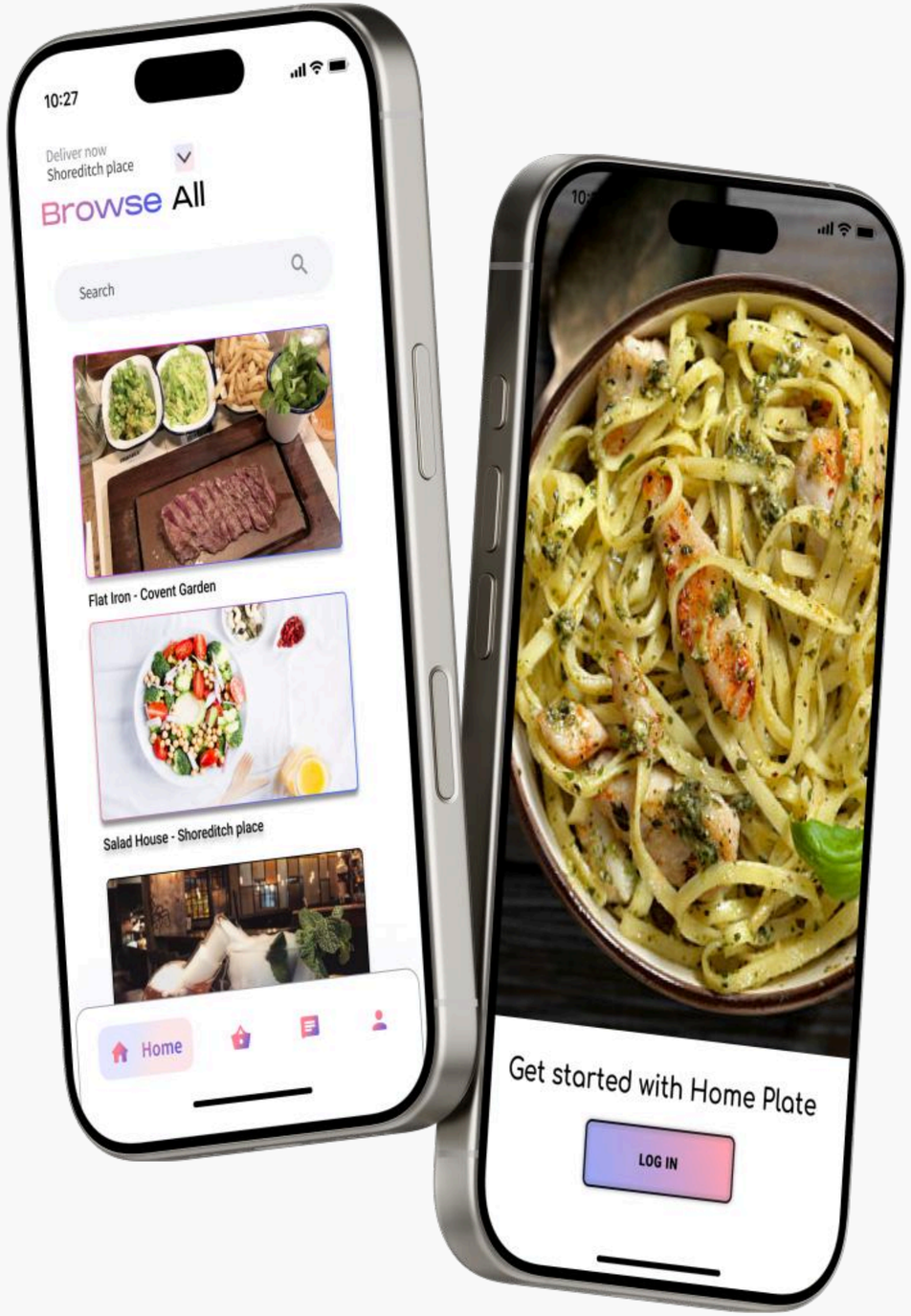
Figure 1

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);

    ImageButton saladhouseButton = findViewById(R.id.imageButton8);
    saladhouseButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View View) {
            startActivity(new Intent( packageContext: HomeActivity.this, menuscreen.class));
        }
    });

    ImageButton flatironButton = findViewById(R.id.imageButton7);
    flatironButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View View) {
            startActivity(new Intent( packageContext: HomeActivity.this, menuscreen.class));
        }
    });

    ImageButton alchemistButton = findViewById(R.id.imageButton9);
    alchemistButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View View) {
            startActivity(new Intent( packageContext: HomeActivity.this, menuscreen.class));
        }
    });
}
```

Android Studio Flamingo

Moving parallel with design and code, the last stop in Android Studio was the kitchen screen or “HomeActivity” as mentioned earlier. Thus, a new activity called “menuscreen” was made to put some dish options for the users to choose from the kitchens. As mentioned on Figma and earlier all kitchen will have the same menu to keep the explanation simple for the documentation. Figure 1 shows every button for every kitchen displayed in figure 9. Demonstrated in Figure 1 “saladhouseButton” the button responsible for salad house’s functionality moving it from the kitchen screen or “HomeActivity” to “menuscreen” when clicked. This will allow the user to move to the next step of the project. ‘menuscreen’ will have bottom navigation bar to easily navigate through the mobile application and display where the user is when navigating through. Next button was “flatironButton” the image button functionality is moved from the kitchen screen or "HomeActivity" to the "menuscreen". The last button is “AlchemistButton” an image button coded to take the user from "HomeActivity" to all same "menuscreen”. Therefore, the “HomeActivity” is all built and ready to be used.





Project

→

Queuer

For this project, It was tasked with designing a series of user interface screens to promote a new mobile app. It developed range of UX/UI elements including wireframes, user flow diagrams, and interactive prototypes. Throughout the process, It collaborated closely with the client to ensure the designs effectively conveyed the app's key features and maintained brand consistency across all digital touchpoints.

Queuer purpose target is too ensure customer a ready to get in Queue at any commercial space. This idea was built to service the community a great opportunity to stand in the Queue when the Queue is nearly done and the customer is upcoming next.

Project

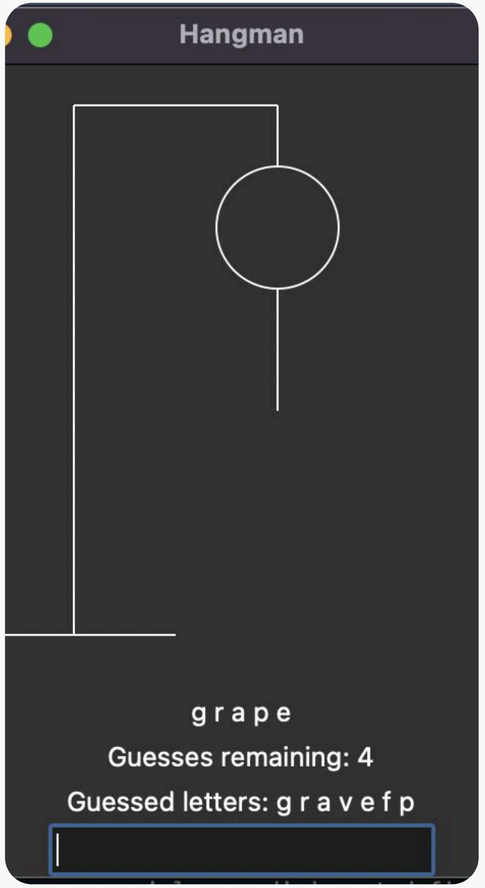
Python Programs

→

1. Python Hangman
2. Python Dataset
3. Python Calculator
4. Python Password Generator

Project

→



Python Hangman

For this project, development was Python-based Hangman game as an engaging and interactive application. The game features a user-friendly interface, customizable word lists, and dynamic feedback to enhance the player's experience. Designed with efficient coding practices, the project showcases logical structuring, string manipulation, and conditional programming, making it both an educational and entertaining tool for users.

```python
import tkinter as tk
from tkinter import messagebox
import random


# Define the list of words
words = ["apple", "banana", "cherry", "orange", "lemon", "grape", "pineapple", "watermelon"]

# Choose a random word from the list
word = random.choice(words)

# Initialize the number of guesses and the list of guessed letters
guesses = 6
guessed_letters = []

# Create the main window
root = tk.Tk()
root.title("Hangman")

# Create the canvas for drawing the hangman
canvas = tk.Canvas(root, width=300, height=300)
canvas.grid(column=0, row=0)

# Draw the scaffold
canvas.create_line(20, 280, 120, 280)
canvas.create_line(70, 280, 70, 20)
canvas.create_line(70, 20, 170, 20)
canvas.create_line(170, 20, 170, 50)

# Create a label for displaying the word
word_label = tk.Label(root, text=" ".join(["_" for letter in word]))
word_label.grid(column=0, row=1)

# Create a label for displaying the number of guesses remaining
guesses_label = tk.Label(root, text="Guesses remaining: {}".format(guesses))
guesses_label.grid(column=0, row=2)

# Create a label for displaying the letters guessed so far
guessed_label = tk.Label(root, text="Guessed letters: ")
guessed_label.grid(column=0, row=3)

# Create an entry for the user to guess a letter
guess_entry = tk.Entry(root)
guess_entry.grid(column=0, row=4)
```

```python
# Define a function to check the user's guess
def check_guess():
    global guesses
    global guessed_letters
    global word_label

    guess = guess_entry.get().lower()
    guess_entry.delete(0, tk.END)

    # Check if the guess is a single letter
    if len(guess) != 1 or not guess.isalpha():
        return

    # Check if the guess has already been guessed
    if guess in guessed_letters:
        return

    guessed_letters.append(guess)
    guessed_label.config(text="Guessed letters: {}".format(" ".join(guessed_letters)))

    # Check if the guess is in the word
    if guess in word:
        word_list = list(word_label["text"])
        for i in range(len(word)):
            if word[i] == guess:
                word_list[2*i] = guess
        word_label.config(text="".join(word_list))

        # Check if the user has won
        if "_" not in word_list:
            playsound("win.wav")
            messagebox.showinfo("Hangman", "You win!")
            retry_button.grid(column=0, row=5)
            exit_button.grid(column=0, row=6)
            guess_entry.config(state=tk.DISABLED)
            return

    # If the guess is not in the word, decrement the number of guesses remaining
    else:
        guesses -= 1
        guesses_label.config(text="Guesses remaining: {}".format(guesses))

        # Draw the hangman
        if guesses == 5:
```

Python Heart Disease Dataset

```
#it prints out the summary of the dataset to tell you if there's any empty values - which there isnt
df.info()
```

✓ 0.2s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
```

Attribute Information

- Age: age of the patient [years]
- Sex: sex of the patient [M: Male, F: Female]
- ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- RestingBP: resting blood pressure [mm Hg]
- Cholesterol: serum cholesterol [mg/dl]
- FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
- RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
- ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
- Oldpeak: oldpeak = ST [Numeric value measured in depression]
- ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- HeartDisease: output class [1: heart disease, 0: Normal]

```
df.head(10)
```

✓ 0.2s

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
5	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0
6	45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
7	54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
8	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1

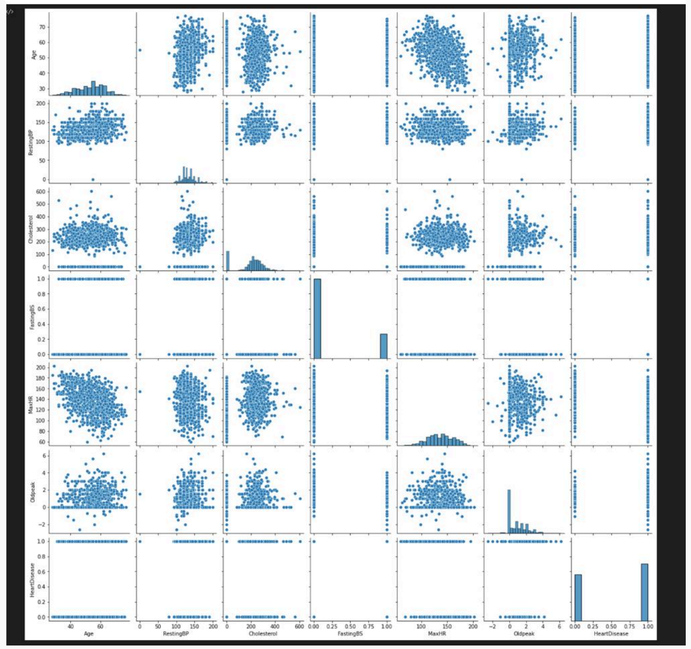
This screenshot shows more info about the dataset as stated “df.info”. It gives an output of the dataset to tell you if there’s any empty values. That’s why df.info() was used to give us the info about the elements used to identify the entitys’ cases. The word null means nothing. So, in that instance non-null means the opposite of nothing. That indicates that there is an entity for every single element used to identify the entitys or people.

It is displayed to the part where “df” is defined in the screenshot. For that particular reason in that circumstance Pandas package was used after defining df. Also, used to read the csv which is the dataset. After then comes the then line df.head(10) which was used to output the first 10 rows and was requested between the brackets. Moreover, Head was the command the requests ‘the first’.

Python Heart Disease Dataset

```
# correlation matrix
plt.figure(figsize=(20,10))
c= df.corr()
sns.heatmap(c,cmap = "PiYG",annot=True)
✓ 0.2s
AxesSubplot:>
```

Plots of Correlation:



The correlation matrix is used as a heat map for defining the intensity heartdisease, restingBP, FastingBS, MaxHR, etc. as stated. Seaborn package was used as SNS to put in a heat map as shown in the code as well as for every element that existed or “True”.

Since it’s a binary classification problem, an algorithm is needed that is able to classify datasets and predict whether a patient has heart disease or not. Therefore, this paper will explore using two classifying algorithms – logistic regression and random forest – to see which performs better. To evaluate the model we can use performance metrics such as accuracy and recalls.

Furthermore, these plots of correlation are in pairs that investigates correlated features. It doesn’t show like there’s any really strong correlation, therefore it should be all right to use a regression.

Python Heart Disease Dataset

```
arn.metrics import confusion_matrix, classification_report
earn.linear_model import LogisticRegression
earn.model_selection import train_test_split


re the columns that have text values. We need to transform them by 'getting dummies' to make the columns numerical
egorical = ['Sex','ChestPainType', 'ExerciseAngina', 'ST_Slope', 'RestingECG']

taframe stored the columns that the model is trained on
rop(columns = ['HeartDisease'])

taframe stores the target variable - or the heart disease column - which says if ppl have heart disease or not
'HeartDisease']].iloc[:,0]

 the bit where we get dummies to make sure all of the columns are numerical
sed = pd.get_dummies(X, columns = cols_categorical, drop_first=True)

 bit we split the full dataset into the testing and training dataset. The training dataset is used for creating the model and
t dataset is used for evaluating the model and getting the accuracy score
X_test, y_train, y_test = train_test_split(X_processed, y, test_size=0.3, random_state=42)
```

Logistic Regression

```
### fit the logistic regression to the training daatset

logModel = LogisticRegression(max_iter=10000)
logModel.fit(X_train, y_train)

y_pred_logmodel = logModel.predict(X_test)

#the classification report shows the accuracy, the precision and the recall
print(classification_report(y_test, y_pred_logmodel))

#this bit prints the confusion matrix
print(confusion_matrix(y_test, y_pred_logmodel))
```

✓ 0.4s

	precision	recall	f1-score	support
0	0.85	0.89	0.87	112
1	0.92	0.89	0.91	164
accuracy			0.89	276
macro avg	0.89	0.89	0.89	276
weighted avg	0.89	0.89	0.89	276

```
[[100  12]
 [ 18 146]]
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier


##### fit the random forest to the training dataset

randforest_model = RandomForestClassifier()

randforest_model.fit(X_train, y_train)

y_pred_rf = randforest_model.predict(X_test)

#the classification report shows the accuracy, the precision and the recall
print(classification_report(y_test, y_pred_rf))

#this bit prints the confusion matrix
print(confusion_matrix(y_test, y_pred_rf))
```

	precision	recall	f1-score	support
0	0.86	0.85	0.85	112
1	0.90	0.90	0.90	164
accuracy			0.88	276
macro avg	0.88	0.88	0.88	276
weighted avg	0.88	0.88	0.88	276

Sklearn is a package we used for the metrics and linear model. From Sklearn we imported more methods to be used for commands and data ouput. In cols_categorial we transformed the strings into integers to identify 1 or 0 which represents True or False. Df and its commands are more used for the heartdisease area to see who is true or false.

This conclusion actively demonstrates that the accuracy of results was 89% as demonstrated. In the f1- score rows it shows the accuracy of 0.89. Precisely, 100 was 12 was wrong and 18 was wrong and 146 was right which makes the logistic regression accurate enough to be used.

Random forest was imported and its another method of evaluating the accuracy of the dataset. At this example the accuracy is 88% as shown like stated above. With f1-score and accuracy column is the output of the code that basically shows the Random Forest method output. As comes the input “x train and y train” rain forest commands is used to use the method. And shown above every comment explains the code.

```
    print("Welcome to the Python Calculator!")
    print("Select operation:")
    print("1. Addition (+)")
    print("2. Subtraction (-)")
    print("3. Multiplication (*)")
    print("4. Division (/)")
    print("5. Quit")

    while True:
        # Get user choice
        choice = input("Enter choice (1/2/3/4/5): ")

        if choice == '5':
            print("Goodbye!")
            break

        if choice in ['1', '2', '3', '4']:
            try:
                # Get numbers from the user
                num1 = float(input("Enter first number: "))
                num2 = float(input("Enter second number: "))

                # Perform the operation
                if choice == '1':
                    print(f"The result of {num1} + {num2} is {num1 + num2}")
                elif choice == '2':
                    print(f"The result of {num1} - {num2} is {num1 - num2}")
                elif choice == '3':
                    print(f"The result of {num1} * {num2} is {num1 * num2}")
                elif choice == '4':
                    if num2 != 0:
                        print(f"The result of {num1} / {num2} is {num1 / num2}")
                    else:
                        print("Error: Division by zero is not allowed.")
            except ValueError:
                print("Invalid input! Please enter numeric values.")
        else:
            print("Invalid choice. Please select a valid operation.")

# Run the calculator
calculator()
```

Project

Python Calculator

```
Welcome to the Python Calculator!
Select operation:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Quit
Enter choice (1/2/3/4/5): 3
Enter first number: 144
Enter second number: 144
The result of 144.0 * 144.0 is 20736.0
Enter choice (1/2/3/4/5):
```

- Based on the user's choice, it performs corresponding operation:
- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/) (with an extra check for division by zero).
- The result is calculated and displayed using print().

This section greets the user and displays the options they can choose from. The operations are numbered (1-5) for easy selection.

- The input() function pauses the program and waits for the user to type something. Whatever the user enters is stored as a string in the variable choice.
- For example, if the user types 1, the choice variable will now hold the value '1'.
- Purpose: This input determines which operation the calculator will perform.

- Prompts the user to input the numbers for the calculation.
- input() takes the user's input as a string, and float() converts it to a number with decimals (if applicable).
  - Example: If the user types 3.5, num1 will hold the value 3.5.
- Questions asked:
  - "Enter first number:" – Asks for the first number.
  - "Enter second number:" – Asks for the second number.
- Using float allows the calculator to handle both integers (e.g., 5) and decimals (e.g., 3.14).

Project

→

```python
import random
import string

def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password

# Ask the user for the desired password length
length = int(input("Enter the desired password length: "))
print("Your generated password is:", generate_password(length))
```

Python Password Generator

```
In [2]: %runfile /Users/alitarek/untitled1.py --wdir
Enter the desired password length: 8
Your generated password is: hOag$%Cr
```

How It Works:

Modules Used:

- random: For random selection.
- string: Provides predefined sets of characters (letters, digits, and punctuation).

Password Composition:

- Combines all possible characters (uppercase, lowercase, digits, special characters) using string.ascii_letters, string.digits, and string.punctuation.

Password Generation:

- Uses random.choice() to select random characters from the combined set.
- Repeats the process for the specified length.

User Input:

- Prompts the user to enter the desired password length.

EXAMPLE ABOVE:

You can customize this further, such as limiting the type of characters or adding specific rules.

GET IN-TOUCH

129-132 London Road, Brighton,UK

+201200992005

alitarek286@gmail.com

GitHub

LINKED IN

Thank you