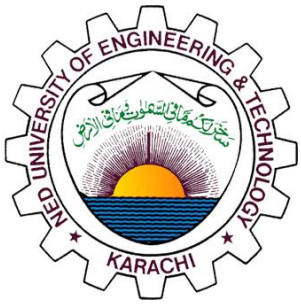
DATABASE MANAGEMENT SYSTEM:



FINAL PROJECT REPORT

PROJECT TITLE: *ONLINE BAKERY SYSTEM*

GROUP MEMBERS:

ZAINAB MOHSIN: CS-24051

BATOOL ZEHRA: CS-24056

MAHNOOR BAIG: CS-24057

BATCH: 2024

SUBMISSION DATE: 25-NOVEMBER-2024

SUBMITTED TO: DR MARIA WAQAS & MISS MEHWISH RAZA

ACKNOWLEDGEMENT:

We would like to express our heartfelt gratitude to our course instructor **DR MARIA WAQAS** and our practical teacher **MISS MEHWISH RAZA** for their guidance, support and cooperation. Their guidance helped us to create this successful database management project. Also we would like to acknowledge the support and contribution of our group members our collective efforts played an important in the execution of this project.

ABSTRACT:

This online bakery system project requires the development of scalable database to manage information relates to bakery products, their availability, & price. This database also store customer details like, name, contact number, email address, home address, and payment status and payment method. This database will prioritize data integrity & offer real time access to authorized users.

This database will also serve as foundation of online bakery system that facilitates efficient management and maintenance of bakery data.

Report on the Online Bakery System Database:

Conceptualization:

IDENTIFYING ENTITIES:

Many websites were visited, and numerous articles were read to gather information about various entities relevant to the Online Bakery Services project. This research was crucial in shaping the concept and selecting entities that best suit the needs of our project. Specifically tailored to the requirements of **craving corner** bakery's online bakery services, the chosen entities were carefully selected to ensure they align perfectly with the project's objectives and operational scope:

- Customer basic information
- Customer address
- About the order
- inventory

Fig 1

Fig 2

DISTINGUISHING FEATURES USED IN PROJECT:

1. **newline=”** : The newline=" argument is used in **file handling** functions, specifically in the open() function when opening a file for reading or writing. It controls how newlines (\n) are handled in text files.
2. **next() :** It is a **built-in** function that is commonly used to iterate through items in an iterator(e.g list, tuple etc) one by one.
3. **enumerate()** : It is a **built-in** function that adds a counter (or index) to an iterable, such as a list, tuple, or string.
4. **os.replace() :** The os.replace() function is used to **replace** a file or directory with another, essentially renaming or overwriting an existing file or directory with a new one. This function is part of the os module, which provides a way to interact with the operating system.
5. **os():** It is **not** a built-in function. However, you might be referring to the **os module**, which is a standard library in Python that provides a way to interact with the operating system.

DATABASE MANAGEMENT WORKFLOW:

The system supports the following core database operations:

- **Creating a New Database:** While the createdatabase() function is referenced in the code, it is not implemented in the provided snippet. This function would likely allow the user to create a new CSV database from scratch.
- **Opening an Existing Database:** The open_atabase() function, also referenced but not implemented, would prompt the user to specify an existing CSV database file to open. Once a database is opened, the user can perform various operations like adding, deleting, searching, or viewing records.
- **Adding a Record:** This functionality would allow the user to input new data, which is then added to the CSV file (though the exact implementation of add_record() is not included in the code).
- **Deleting a Record:** The function delete_field_value(db_name) allows users to modify specific field values in the database (i.e., clearing a field value based on a selected search criterion).
- **Searching a Record:** The search_record() function provides a simple way for the user to search through the database based on the contents of a record. It displays all records containing the search term.
- **Displaying All Records:** The display_records() function simply outputs all the records in the database, giving the user a quick overview of all data entries.

OVERVIEW:

This Python program simulates the management of an online bakery system using a CSV-based database. The system allows users to:

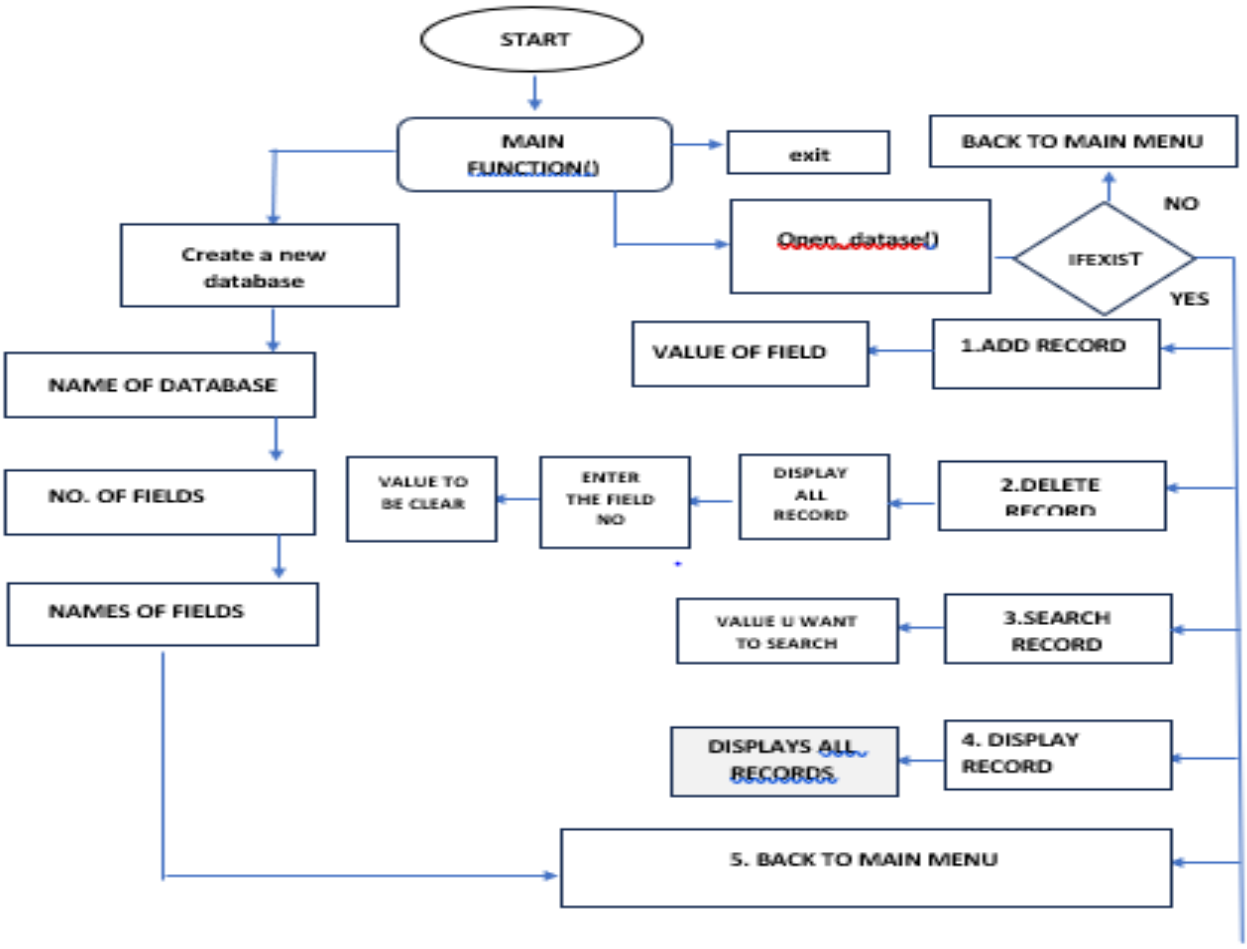
- Create a new database with customizable fields.
- Open an existing database and perform various operations.
- Add, delete, and search records in the database.
- Display all records or specific data within a database.

The program works interactively through the terminal or command-line interface, guiding users to perform actions like adding new records, searching for specific data, and deleting records. The data for the bakery (or any other type) is stored in CSV files, which are easy to handle, edit, and export.

FLOWCHARTS OF OUR PROJECT:

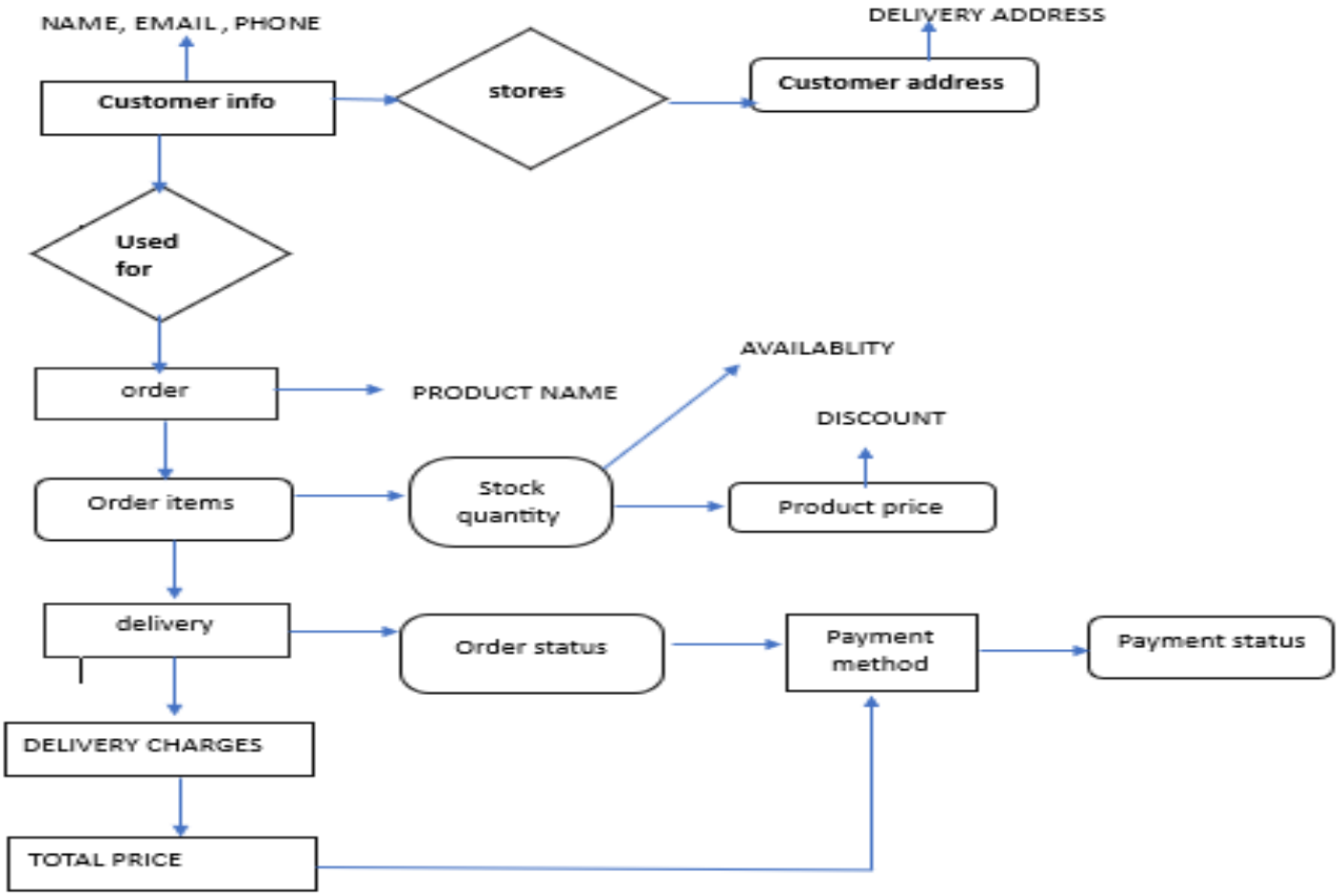
FLOWCHART NO: 1

FLOWCHART OF THE PROGRAM:



FLOWCHART NO: 2

Flowchart of the database (ONLINEBAKERYSYSTEM):



MOST CHALLENGING PART:

The most challenging part while working on the project were as follows

- Correctly handing file, i.e. Opening, reading, writing, and modifying CSV files
- Modifying records, Deleting, adding, or searching for specific records without causing data loss
- Usage of various distinguishing features such as **enumerate, os.replace(), next()**

NEW THING LEARNT WHILE WORKING ON PROJECT:

- File Handling (Opening, Writing, Reading, and Modifying CSV Files)
- Managing modifications in a **temporary file** (like `temp.csv` in our code)
- **Debugging** skills for catching and fixing errors
- Working with data structures like **list** and **strings**

REFERENCES:

- Chatgpt
- YouTube
- Seniors

DESCRIPTION OF THE FUNCTIONS USED I.E.:

create_database (), open_database (), add_record (), delete_field_value (), search_record () and display_record ():

OVERVIEW:

The following functions allow the user to interact with the database by creating a new one, opening an existing database, adding records to the database, deleting field values, searching records within database and displaying records of database. These functions form the core of the program, enabling database creation.

1. Function: create_database ()

Purpose:

The create_database () function allows the user to create a new CSV database. It prompts the user for a name for the database file and defines the fields (columns) that will structure the data in the database.

Workflow:

- **Prompt for Database Name**: The user is asked to enter a name for the new database. The function appends the .csv extension to the input to create a file name for the database.
- **Prompt for Number of Fields**: The function requests the user to specify the number of fields (columns) for the database. If the user enters an invalid number (non-integer), the function catches the error and asks the user to input a valid integer.
- **Gather Field Names**: The user is prompted to enter names for each field, and the function stores these names in a list.
- **Write to CSV File**: Once the field names are collected, the function creates and opens a new CSV file. It writes the field names as the header row in the file, marking the structure of the database

EXAMPLE:

The name for the new database: ONLINE BAKERY SYSTEM

Enter the number of fields: 15

Enter name of field 1: S.NO

Enter name of field 2: FIRST NAME

Enter name of field 3: LAST NAME

Enter name of field 4: PHONE NUMBER

Enter name of field 5: EMAIL ADDRESS

Enter name of field 6: DELIVERY ADDRESS

Enter name of field 7: PRODUCT NAME

Enter name of field 8: PRODUCT PRICE

Enter name of field 9: AVAILABIITY

Enter name of field 10: DELIVERY CHARGES

Enter name of field 11: DISCOUNT%

Enter name of field 12: TOTAL PRICE

Enter name of field 13: PAYMENT METHOD

Enter name of field 14: PAYMENT STATUS

Enter name of field 15: ORDER STATUS

Database 'onlinebakerysystem.csv' created successfully.

Key Points:

- The CSV file is created with headers corresponding to the field names provided by the user.
- The database file is stored in the same directory as the script.
- The user is guided through the creation process with appropriate error handling for invalid input.

2. Function: open_database ()

Purpose:

The open_database () function allows the user to open an existing database by specifying its name. It checks if the database exists before opening it.

Workflow:

- **Prompt for Database Name**: The user is asked to input the name of an existing database. The function appends the .csv extension to the name to form the complete file name.
- **Check Database Existence**: The function checks if the database file exists in the current directory using the os.path.exists() method.
    - **If the file exists**: The function prints a success message and returns the database name to be used for further operations.
    - **If the file does not exist**: The function notifies the user that the database could not be found and returns none, signaling an error in opening the database.

Example Interaction:

Enter the name of the existing database: ONLINE BAKERY SYSTEM

Database 'onlinebakerysystem.csv' opened.

Key Points:

- The function only allows interaction with an existing database and ensures the file's existence.
- If the user attempts to open a non-existent file, the program prevents further actions and informs the user of the error.
- It returns the database name if successful, which is used in subsequent database operations (such as adding records).

3. Function: add_record ()

Purpose:

The add_record() function allows the user to insert a new record into the open database. It prompts the user for values for each field defined in the database and appends the new record to the CSV file.

Workflow:

- **Open the CSV File**: The function opens the specified CSV file in append mode ('a+'), allowing it to add data to the end of the file without overwriting existing records.
- **Read Field Names**: The function reads the header (field names) from the file to determine the structure of the data. If the database file is empty or does not contain any records, an error is shown, and the function exits.
- **Prompt for Field Values**: For each field in the database, the function prompts the user to enter a value. These values are collected into a list that represents the new record.
- **Write Record to CSV**: Once the data for all fields has been collected, the function appends the new record (a list of field values) as a new row in the CSV file.
- **Feedback**: After the record is added, the function prints a success message confirming that the record has been inserted.

EXAMPLE:

Enter value for FIRST NAME: Rameen

Enter value for LAST NAME: Sohail

Enter value for PHONE NUMBER: 923566728928

Enter value for EMAIL ADDRESS: rami@gmail.com

Enter value for DELIVERY ADDRESS: gulshan, sector33

Enter value for PRODUCT NAME: doughnuts

Enter value for PRODUCT PRICE: 300

Enter value for AVAILABILITY: yes

Enter value for DELIVERY CHARGES: 100

Enter value for DISCOUNT%: -

Enter value for TOTAL PRICE: 400

Enter value for PAYMENT METHOD: ONLINE

Enter value for PAYMENT STATUS: UNPAID

Enter value for ORDER STATUS: PENDING

Record added successfully.

Key Points:

- The function reads the header row to determine the structure of the database and ensures that data is inserted correctly.
- Each record is appended to the database without affecting existing records.
- If the database is empty (i.e., no header row exists), the function displays an error message and terminates the record addition process.
- The function provides feedback to the user, confirming the successful addition of the record.

The create_database(), open_database(), and add_record() functions form the essential components of the database management system in this script. These functions enable the user to create a new database, open an existing one, and insert records, providing a foundation for further operations such as deleting, searching, and displaying records. The user-friendly prompts and error handling ensure that the process is straightforward, while the feedback messages keep the user informed of each operation's success or failure.

4. Function: delete_field_value()

The delete_field_value function is designed to allow users to modify a specific record in a CSV-based database by clearing the value of a particular field. It ensures user interaction and verification, as well as careful handling of data to avoid accidental loss. Below is a detailed breakdown of the function's behavior:

- **Displaying Current Records**

The function starts by displaying the current records in the database using the display_records function. This allows the user to review the existing data before making any changes. This is a critical first step in ensuring that the user is aware of the current state of the database.

- **Opening the CSV File**

The function then opens the CSV file in read mode ('r') to read its contents. The csv.reader is used to parse the file, and all rows are loaded into a list. This enables easy manipulation of the data later in the function. If the CSV contains only the header (i.e., no records), the function immediately informs the user that no records are available to modify and exits gracefully.

- **Choosing the Field to Modify**

Next, the function prompts the user to select which field they want to modify. This is achieved by displaying a numbered list of field names from the header row of the CSV file. The user is then asked to enter the corresponding number for the field they wish to change.

- **Input Validation:**
  - The user is asked to enter a number corresponding to one of the fields. If the input is invalid (i.e., not a valid number or out of range), the function notifies the user and terminates.

- **Selecting the Record to Modify**

The function then prompts the user to specify the value of the chosen field for which they wish to clear the data. This step ensures that the user can target a specific record by matching the field value.

- **Modifying the Record**
    - A temporary file (temp.csv) is created to store the modified database. The function loops through all rows of the original CSV, copying them over to the temporary file.
    - If a row matches the user's specified value for the selected field, that field is cleared (set to an empty string). The modification flag is set to True to indicate that a change was made.
    - After processing all rows, the modified data in the temporary file replaces the original CSV file.

- **Completion:**

Once the modification is complete, the function checks if any changes were made:

    - If a modification was made, the function confirms the successful clearing of the field.
    - If no matching record was found to modify, the function informs the user that no changes were made.
- **Edge Cases and Error Handling**
    - **No Records:** If the CSV contains only the header (no records), the function gracefully exits and informs the user.
    - **Invalid Field Number:** If the user inputs a field number that is out of range or invalid, the function terminates with an appropriate error message.
    - **No Matching Record:** If no record matches the specified field value, the function lets the user know that no modification was made.
- **Separator Line**

After completion, the function prints a separator line for clarity and formatting purposes, ensuring that the user can easily distinguish between different operations in the console.

Limitations:

- The function currently only clears a field value, rather than deleting an entire record or modifying multiple fields at once.
- It assumes the database is stored in CSV format and does not support other formats (e.g., SQL databases or JSON).

  The delete_field_value function is a user-friendly tool for selectively modifying records in a CSV file. It provides clear guidance to the user while ensuring that the integrity of the data is maintained throughout the process. However, it could be extended to allow for more flexible operations, such as deleting entire records or modifying multiple fields at once.

5. Function: search_record()

- **Purpose:** This function allows the user to search for specific records within a CSV database.
- **Process:**
    - The user is prompted to enter a value to search for in the database.
    - The function reads the CSV file and checks each row for the search value.
    - If a match is found, it prints the matching record. If no match is found, it notifies the user with a message: ! Record not found.
- **User Interaction:** Clear prompts guide the user through entering a search value. If no matches are found, the user is informed accordingly.

6. Function: display_records()

- **Purpose:** Displays all records in the selected database.
- **Process:**
    - The function reads the entire database file, checks if there are records (beyond the header), and displays them.
    - If no records exist (i.e., only the header is present), the function informs the user with a message: ! No records to display.
- **User Interaction:** This function provides a complete list of records for the user to review, with a helpful message if no records are available.

7. Function: main()

- **Purpose:** Acts as the entry point for the program and controls the flow of user interaction with the system.
- **Process:**
    - The function presents a main menu with options to either create a new database, open an existing one, or exit the program.
    - If the user chooses to open an existing database, a secondary menu is displayed that allows them to add, delete, search, or display records within the selected database.
    - The menu system uses a loop to repeatedly display options until the user chooses to exit or return to the previous menu.
- **User Interaction:** The main function provides a structured interface, guiding the user through various actions such as selecting a database or performing operations like adding and searching records.

OUTPUTS:

FIG NO 3: Output of **(ONLINEBAKERYSYSTEM)** Database.

```
 Displaying all records:
['S.NO', 'FIRSTNAME', 'LASTNAME', 'PHONE NO', 'EMAIL ADDRESS', 'DELIVERY ADDRESS', 'PRODUCT NAME', 'PRODUCT PRICE', 'AVAILABILITY', 'DELIVERY CHARGES', 'D
['1', 'Rameen', 'sohail', '923566728928', 'rami@gmail.com', 'gulshan,sector33', 'doughnuts', '300', 'yes', '100', '-', '400', 'online', 'unpaid', 'pending
['2', 'Aliza', 'zai', '923678399202', 'aliaz@gmail.com', 'korangi', 'cake', '700', 'yes', '200', '-', '900', 'cash on delivery', 'unpaid', 'pending']
['3', 'Ali', 'baig', '9328992032233', 'Alibg@gmail.com', 'waterpump,sector33', 'Cake', '500', 'yes', '150', '-', '650', 'online', 'unpaid', 'pending']
['4', 'Umer', 'Khan', '9266738892244', 'Umh@gmail.com', 'gulshan', 'shake', '300', 'yes', '100', '-', '400', 'online', 'paid', 'delivered']
['5', 'marib', 'khan', '9236637782663', 'maribk@gmail.com', 'northnazimabad,sector b', 'buns', '400', 'yes', '100', '-', '500', 'online', 'unpaid', 'pendi
['6', 'fatima', 'sohail', '9255637728883', 'femi@gmail.com', 'saima villas block h', 'muffins', '600', 'yes', '100', '-', '700', 'online', 'paid', 'pendir
['7', 'ajhar sheikh', 'sheikh', '924435562672', 'lolhigh@gmail.com', 'newkarachi,block b', 'pastries', '700', 'yes', '300', '-', '1000', 'online', 'paid',
['8', 'umama', 'khan', '925563736283', 'umamakhan45@gmail.com', 'gulshan sector77', 'chocolate brownies', '500', 'yes', '150', '-', '650', 'online', 'paid
['9', 'yusra', 'azeem', '923456778961', 'yusi@gmail.com', 'saima villas block c', 'croissant', '700', 'yes', '100', '-', '800', 'online', 'unpaid', 'pendi
['9', 'umaima ', 'khuzaima', '9266738893744', 'khum@gmail.com', 'north nazimabad', 'pie', '200', 'yes', '100', '-', '300', 'online', 'paid', 'pending']
['10', 'arbish', 'azeem', '926733782937', 'arb@gamil.com', 'waterpump,sector 1', 'chocolate vars', '400', 'yes', '100', '-', '500', 'online', 'unpaid', 'p
['11', 'daniyal', 'bhaawgrah', '926673882636', 'dani2gmail.com', 'university road', 'sandwiches', '150', 'yes', '100', '-', '270', 'online', 'paid', 'pend
['12', 'Saira', 'arif', '9244516626667', 'sairaarif2gmail.com', 'hussainabad', 'graham crackers', '300', 'yes', '200', '-', '500', 'online', 'paid', 'on t
['13', 'barza', 'bilal', '924453662744', 'barz@gamil.com', 'northkarachi,sector 2', 'galette and toasts', '1500', 'yes', '200', '-', '1700', 'online', 'pa
--------------------------------------------------------------------------------------------------
```

FIG NO 4:

```
S.NO,FIRSTNAME,LASTNAME,PHONE NO,EMAIL ADDRESS,DELIVERY ADDRESS,PRODUCT NAME,PRODUCT PRICE,AVAILABILITY,DELIVERY CHARGES,DISCOUNT%,TOTAL
1,Rameen,sohail,923566728928,rami@gmail.com,"gulshan,sector33",doughnuts,300,yes,100,-,400,online,unpaid,pending
2,Aliza,zai,923678399202,aliaz@gmail.com,korangi,cake,700,yes,200,-,900,cash on delivery,unpaid,pending
3,Ali,baig,9328992032233,Alibg@gmail.com,"waterpump,sector33",Cake,500,yes,150,-,650,online,unpaid,pending
4,Umer,Khan,9266738892244,Umh@gmail.com,gulshan,shake,300,yes,100,-,400,online,paid,delivered
5,marib,khan,9236637782663,maribk@gmail.com,"northnazimabad,sector b",buns,400,yes,100,-,500,online,unpaid,pending
6,fatima,sohail,9255637728883,femi@gmail.com,saima villas block h,muffins,600,yes,100,-,700,online,paid,pending
7,ajhar sheikh,sheikh,924435562672,lolhigh@gmail.com,"newkarachi,block b",pastries,700,yes,300,-,1000,online,paid,delivered
8,umama,khan,925563736283,umamakhan45@gmail.com,gulshan sector77,chocolate brownies,500,yes,150,-,650,online,paid,delivered
9,yusra,azeem,923456778961,yusi@gmail.com,saima villas block c,croissant,700,yes,100,-,800,online,unpaid,pending
9,umaima ,khuzaima,9266738893744,khum@gmail.com,north nazimabad,pie,200,yes,100,-,300,online,paid,pending
10,arbish,azeem,926733782937,arb@gamil.com,"waterpump,sector 1",chocolate vars,400,yes,100,-,500,online,unpaid,pending
11,daniyal,bhaawgrah,926673882636,dani2gmail.com,university road,sandwiches,150,yes,100,-,270,online,paid,pending
12,Saira,arif,9244516626667,sairaarif2gmail.com,hussainabad,graham crackers,300,yes,200,-,500,online,paid,on the way
13,barza,bilal,924453662744,barz@gamil.com,"northkarachi,sector 2",galette and toasts,1500,yes,200,-,1700,online,paid,delivered
```

FIG NO 5:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	S.NO	FIRSTNAME	LASTNAME	PHONE NO	EMAIL ADDRESS	DELIVERY ADDRE	PRODUCT NAME	PRODUCT PRICE	AVAILABILITY	DELIVERY CHARG	DISCOUNT%	TOTAL PRICE	PAYMENT METH	PAYMENT STATU	ORDER STATUS
2	1	Rameen	sohail	923566728928	rami@gmail.co	gulshan,sector3	doughnuts	300	yes	100	-	400	online	unpaid	pending
3	2	Aliza	zai	923678399202	aliaz@gmail.co	korangi	cake	700	yes	200	-	900	cash on deliver	unpaid	pending
4	3	Ali	baig	9328992032233	Alibg@gmail.co	waterpump,sect	Cake	500	yes	150	-	650	online	unpaid	pending
5	4	Umer	Khan	9266738892244	Umh@gmail.co	gulshan	shake	300	yes	100	-	400	online	paid	delivered
6	5	marib	khan	9236637782663	maribk@gmail.	northnazimaba	buns	400	yes	100	-	500	online	unpaid	pending
7	6	fatima	sohail	9255637728883	femi@gmail.co	saima villas blo	muffins	600	yes	100	-	700	online	paid	pending
8	7	ajhar sheikh	sheikh	924435562672	lolhigh@gmail.	newkarachi,blo	pastries	700	yes	300	-	1000	online	paid	delivered
9	8	umama	khan	925563736283	umamakhan45(	gulshan sector7	chocolate brow	500	yes	150	-	650	online	paid	delivered
10	9	yusra	azeem	923456778961	yusi@gmail.co	saima villas blo	croissant	700	yes	100	-	800	online	unpaid	pending
11	9	umaima	khuzaima	9266738893744	khum@gmail.co	north nazimaba	pie	200	yes	100	-	300	online	paid	pending
12	10	arbish	azeem	926733782937	arb@gamil.com	waterpump,sect	chocolate vars	400	yes	100	-	500	online	unpaid	pending
13	11	daniyal	bhaawgrah	926673882636	dani2gmail.com	university road	sandwiches	150	yes	100	-	270	online	paid	pending
14	12	Saira	arif	9244516626667	sairaarif2gmail	hussainabad	graham cracker	300	yes	200	-	500	online	paid	on the way
15	13	barza	bilal	924453662744	barz@gamil.co	northkarachi,se	galette and toa	1500	yes	200	-	1700	online	paid	delivered
16															

ATLEAST THREE TEST CASE RUNS:

These test cases demonstrate the main functionalities of the program: creating a database, adding records, searching for records, and deleting specific field values.

Test Case 1: Create a New Database and Add a Record

STEPS:

- Create a new database: ONLINEBAKERYSYSTEM
- Enter fields: product, product price

```
PS C:\Users\Hp> & C:/Users/Hp/AppData/Local/Programs/Python/Python312/python.exe c:/cep/neddbpro.py
                      PROGRAM FOR DATA BASE MANAGEMENT SYSTEM

=== Main Menu ===
1. Create a new database
2. Open an existing database
3. Exit
=================
-> Enter your choice: 1
-> Enter the name for the new database: ONLINEBAKERYSYSTEM
-------------------------------------------------------------------------------------------------------
->  Enter the number of fields: 2
-> Enter name of field 1: PRODUCT
-> Enter name of field 2: PRODUCT PRICE

 Database 'ONLINEBAKERYSYSTEM.csv' created successfully.
-------------------------------------------------------------------------------------------------------
```

```
=== Main Menu ===
1. Create a new database
2. Open an existing database
3. Exit
=================
-> Enter your choice: 2
->  Enter the name of the existing database: ONLINEBAKERYSYSTEM
-------------------------------------------------------------------------------------------------------
 Database 'ONLINEBAKERYSYSTEM.csv' opened.
```

- Enter fields: product, product price
- Add a record with values: MILK, 120

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: 1
-> Enter value for PRODUCT: MILK
-> Enter value for PRODUCT PRICE: 120

 Record added successfully.
-----------------------------------------------------------------------------------------------
```

- Display the records

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: 4

 Displaying all records:
['PRODUCT', 'PRODUCT PRICE']
['MILK', '120']
-----------------------------------------------------------------------------------------------
```

Test Case 2: Search for a Record

STEPS:

- Open the existing database ONLINEBAKERYSYSTEM,.csv
- Search for a record with the value for instance MILK, PRODUCT PRICE, providing an error if wrong value is entered.

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: 3
-> Enter the value to search: MILK
Record found: ['MILK', '120']
------------------------------------------------------------------------------------------------------

=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: 3
-> Enter the value to search: 120
Record found: ['MILK', '120']
------------------------------------------------------------------------------------------------------
```

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: 3
-> Enter the value to search: PRODUCT
Record found: ['PRODUCT', 'PRODUCT PRICE']
--------------------------------------------------------------------------------------------------------

=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
====================
-> Enter your choice: PRICE
! Invalid choice. Try again.
```

Test Case 3: Delete a Record's Field Value

STEPS:

- Open the database ONLINEBAKERYSYSTEM.csv
- Delete the PRODUCT value for MILK
- Display all records after modification.
- For returning back to main menu enter 5

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
=====================
-> Enter your choice: 2

 Current records:

 Displaying all records:
['PRODUCT', 'PRODUCT PRICE']
['MILK', '120']
-----------------------------------------------------------------------------------------------------

 Choose the field to clear the value from:
1. PRODUCT
2. PRODUCT PRICE
-> Enter the field number: 1
->  Enter the value of 'PRODUCT' to clear the field value: MILK

 Field value cleared successfully.
-----------------------------------------------------------------------------------------------------
```

```
=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
=====================
-> Enter your choice: 4

 Displaying all records:
['PRODUCT', 'PRODUCT PRICE']
['', '120']
-----------------------------------------------------------------------------------------------------

=== Database Menu ===
1. Add a record
2. Delete a record
3. Search a record
4. Display all records
5. Back to main menu
=====================
-> Enter your choice: 5

=== Main Menu ===
1. Create a new database
2. Open an existing database
3. Exit
==================
-> Enter your choice:
```

OTHER EXAMPLE OF DATABASES:

Example of databases created other than ONLINEBAKERYSYSTEM:-

1. HOSPTIAL DATABASE
2. STUDENT DATABASE
3. THROWBALLSCORE
4. KDRAMA COLLECTION

1-Hospital Database:

```
 Displaying all records:
['s.no', 'name', 'ward', 'doctor', 'condition']
['1', 'mahnooor', '2', 'hamna', 'diagnosed with insomnia']
['2', 'rameen', '6', 'ali', 'high blood pressure']
['3', 'zainab', '4', 'zara', 'typhoid']
['4', 'batool', '5', 'anas', 'sugar']
--------------------------------------------------------------------------------------------------------
```

2-Student Database:

```
 Displaying all records:
['S.NO', 'NAME', 'ROLL NO', 'PERCENTAGE', 'GRADE']
['01', 'Batool', 'cs-056', '88%', 'A+']
['02', 'Mahnoor baig', 'cs-057', '86%', 'A+']
['03', 'Zainab', 'cs-051', '86%', 'A+']
['04', 'Rameen sohail', 'cs-058', '81%', 'A+']
['05', 'Zerwa', 'cs-077', '70%', 'A']
--------------------------------------------------------------------------------------------------------
```

3-Throwballscore Database:

```
 Displaying all records:
['name', 'score', 'passed']
['fatima', '33', 'yes']
['rameen', '28', 'yes']
['zainab', '28', 'yess']
['batool', '29', 'yes']
['rumaisa', '29', 'yes']
['umaima', '30', 'yess']
['umama', '32', 'yes']
['yusra', '34', 'yes']
['barza', '15', 'no']
--------------------------------------------------------------------------------------------------------
```

4-kdramacollection database:

```
 Displaying all records:
['s.no', 'name', 'date', 'rate ']
['1', 'ALL OF US ARE DEAD', '9-Jan', '*********']
['2', 'DUTY AFTER SCHOOL', '10-Oct', '*********']
['3', 'truebeauty', '9-May', '****']
['4', 'moneyheist', '10-Aug', '*********']
['5', 'bloodhounds', '10-Oct', '********']
['6', 'gyeongseoung creature', '10-Oct', '******']
['7', 'sweethome', '10-Sep', '*******']
['8', 'deathgame', '10-Sep', '**********']
['9', 'doona', '2-Jan', '*']
--------------------------------------------------------------------------------------------------------
```

INDIVIDUAL CONTRIBUTION:

ZAINAB MOHSIN	CS-24051	DELETE RECORD/ REPORT/FLOWCHART
BATOOL ZEHRA	CS-24056	SEARCH RECORD/ DISPLAY RECORD /MAIN FUNCTION/ REPORT/QA TESTER
MAHNOOR BAIG	CS-24057	CREATE DATABASE/ OPEN DATABASE/ ADD RECORD(DATABASES)/ FLOWCHART/ REPORT

CONCLUSION:

This code provides a basic yet functional database management system using CSV files. It allows users to create, open, modify, and view records interactively through a simple text-based interface. While the system is minimal, it lays the foundation for more advanced features like record deletion and database creation. With some additional features and improvements, this system could serve as a basic tool for managing structured data in CSV format.