



Project Cover Page

Assignment Title:	MIDTERM PROJECT		
Assignment No:	01	Date of Submission:	9 December 2024
Course Title:	PROGRAMMING IN PYTHON		
Course Code:	CSC4162	Section:	A
Semester:	Fall	2024-25	Course Teacher: DR. ABDUS SALAM

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* Student(s) must complete all details except the faculty use part.

** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 3

No	Name	ID	Program	Signature
1	KHONDOKER MD. SABIT HASAN	21-45306-2	BSc [CSE]	
2	MD. EMAMUL AREFIN ISLAM	22-46608-1	BSc [CSE]	
3	A. M. RAFINUL HUQ	21-45668-3	BSc [CSE]	
4	FAISAL AMIN ABIR	20-43206-1	BSc [CSE]	
5			Choose an item.	
6			Choose an item.	
7			Choose an item.	
8			Choose an item.	
9			Choose an item.	
10			Choose an item.	

Faculty use only

FACULTY COMMENTS	Marks Obtained	
	Total Marks	

Crossword Puzzle Game

A crossword puzzle is a word-based puzzle typically presented as a grid of black and white squares. The objective is to fill the white squares with words that intersect, based on clues provided for "Across" (horizontal) and "Down" (vertical) directions. We have implemented functions such as user authentication, gameplay, score tracking, administrative features, and file-based storage. The game is console-based and implemented in Python.

Features Implemented:

1. Main

The 'Main' function serves as the entry point for the Crossword Game application, managing the overall flow of the program. It provides a menu-driven interface for users and admins, allowing them to navigate various features based on their roles.

Functionality:

1. Welcome Screen: Displays a welcome message and the main menu options.

2. User Login:

- Authenticates users and determines their role (user or admin).
- Redirects users to their respective dashboards upon successful login.

3. User Dashboard:

- Users can:
 - Play the crossword game.
 - View their scores.
 - Change their password.
 - Log out.

4. Admin Dashboard:

- Admins can:
 - View all registered users.
 - View all user scores.
 - Remove a user from the system.
 - Log out.

5. User Registration: Allows new users to register by providing their details.

6. Exit Option: Enables users to exit the application.

```

def Main():
    print("-----Welcome to Crossword Game!!-----")
    choice_1 = True
    while(choice_1):
        Home()
        first = input("Choose an option: ")

        if first == "1":
            validUser, name, email, role = Login(userDataFile)
            if validUser and role == "user":
                print(f"Login successful!, welcome back '{name}'")
                choice_2 = True
                while(choice_2):
                    DashboardUser()
                    second = input("Choose an option: ")

                    if second == "1":
                        PlayGame(email)
                    elif second == "2":
                        IndividualUserScore(email)
                    elif second == "3":
                        ChangePassword(email)
                    elif second == "4":
                        print("Logged out successfully!")
                        choice_2 = False
                    else:
                        print("Invalid Input")
            elif validUser and role == "admin":
                print(f"Admin login successful!, welcome back '{name}'")
                choice_3 = True
                while(choice_3):
                    DashboardAdmin()
                    third = input("Choose an option: ")
                    if third == "1":
                        AllUser()
                    elif third == "2":
                        AllUserScores()
                    elif third == "3":
                        RemoveUser()
                    elif third == "4":
                        print("Logged out successfully!")
                        choice_3 = False
                    else:
                        print("Invalid Input")

            else:
                print("Invalid user! Please try again.")
        elif first == "2":
            if Registration(userDataFile):
                print("Registration successful! You can now log in.")
        elif first == "3":
            print("Exiting the application. Goodbye!")
            choice_1 = False
        else:
            print("Invalid Input")

Main()

```

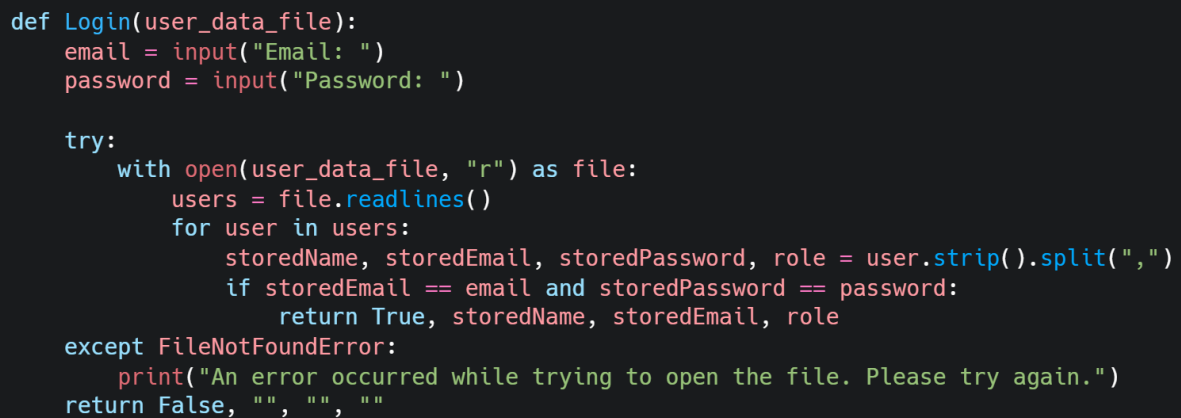
Figure 1: Main Function

2. User Authentication (Login and Registration)

The system provides options for users to log in or register. User credentials are stored in a file named "user_data.txt."

Login Function:

The 'Login' function authenticates users by verifying their email and password against data stored in a file. It prompts the user for their email and password, reads the user data file line by line, and checks for a match. If valid credentials are found, it returns 'True' along with the user's name, email, and role. If the file is missing or credentials are invalid, it returns 'False' with empty values. The function also handles file-related errors gracefully.



```
def Login(user_data_file):
    email = input("Email: ")
    password = input("Password: ")

    try:
        with open(user_data_file, "r") as file:
            users = file.readlines()
            for user in users:
                storedName, storedEmail, storedPassword, role = user.strip().split(",")
                if storedEmail == email and storedPassword == password:
                    return True, storedName, storedEmail, role
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")
    return False, "", "", ""
```

Figure 2: Login Function

Registration Function:

The 'Registration' function registers a new user by saving their name, email, and password to a specified file. It validates the input, checks for duplicate users by comparing the email and password with existing entries, and appends the new user's data if no match is found. If registration is successful, it returns 'True'; otherwise, it returns 'False'. The function also handles file-related errors gracefully.

```

def Registration(user_data_file):
    name = input("Enter your name: ")
    email = input("Enter your valid email Address: ")
    password = input("Enter your password: ")

    name = name.strip()
    email = email.strip()
    password = password.strip()

    if(len(name) == 0 or len(email) == 0 or len(password) == 0):
        print("Invalid input! Please try again.")
        return False
    else:
        try:
            with open(user_data_file, "r") as file:
                users = file.readlines()
                for user in users:
                    storedName, storedEmail, storedPassword, storedRole = user.strip().split(",")
                    if storedEmail == email and storedPassword == password:
                        print("User already exists! Please log in.")
                        return False
        except FileNotFoundError:
            print("An error occurred while trying to open the file. Please try again.")

        try:
            with open(user_data_file, "a") as file:
                file.write(f"{name},{email},{password},user\n")
            return True
        except FileNotFoundError:
            print("An error occurred while trying to open the file. Please try again.")

```

Figure 3: Registration Function

3. Gameplay

The 'PlayGame' function allows users to play a crossword puzzle game. It randomly selects a puzzle from predefined choices and iterates through the clues, prompting the user to answer each. Correct answers earn 10 points, while incorrect answers deduct 5 points. The game can be exited at any time by entering "0". Upon completion or exit, the user's score is saved to a file along with their email and the timestamp. The function also handles file-related errors gracefully.

```

def PlayGame(email):
    pickedPuzzle = random.choice(puzzleChoice)
    i = 0
    score = 0
    while i < len(pickedPuzzle[2]):
        print(pickedPuzzle[0][i])
        print("Hint: ", end="")
        print(pickedPuzzle[1][i])
        print("0. Exit")
        userInput = input("Enter your answer: ")
        if userInput == "0":
            print("Exiting the game. Goodbye!")
            print(f"Your score is {score}")
            break
        elif userInput == pickedPuzzle[2][i]:
            print("Correct!")
            score += 10
            i += 1
            if i == len(pickedPuzzle[2]):
                print(pickedPuzzle[0][i])
                print("Congratulations! You have completed the puzzle.")
                print(f"Your score is {score}")
        else:
            print("Incorrect!")
            score -= 5
            i = i

    try:
        with open(scoreDataFile, "a") as file:
            file.write(f"{email},{score},{datetime.now()}\n")
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")

```

Figure 4: Play Game Function

4. Score Tracking by User

The `IndividualUserScore` function displays all the scores of a specific user identified by their email. It reads the score data file, filters entries matching the provided email, and prints each score and its corresponding date. If the file is missing or cannot be opened, it handles the error gracefully by displaying an appropriate message.

```

def IndividualUserScore(email):
    print(f"The scores of {email} are:")
    try:
        with open(scoreDataFile, "r") as file:
            scores = file.readlines()
            for score in scores:
                userEmail, userScore, date = score.strip().split(",")
                if userEmail == email:
                    print(f"Score: {userScore}; Date: {date}")
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")

```

Figure 5: Individual User Score Function

5. Administrative Features

The system includes an admin role with additional functionalities such as viewing all users, viewing all user scores, and removing users.

Viewing All Users:

The `AllUser` function displays a list of all registered users. It reads the user data file, extracts each user's name, email, and role, and prints them in a formatted manner. If the file is missing or cannot be opened, it handles the error gracefully by displaying an appropriate message.

```

def AllUser():
    print("All User:")
    try:
        with open(userDataFile, "r") as file:
            users = file.readlines()
            for user in users:
                name, email, password, role = user.strip().split(",")
                print(f"Name: {name}; Email: {email}; Role: {role}")
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")

```

Figure 6: All User Function

Viewing All User Scores:

The `AllUserScores` function displays the scores of all users. It reads the score data file, extracts each user's email, score, and the date of the score, and prints them in a formatted manner. If the file is missing or cannot be opened, it handles the error gracefully by displaying an appropriate message.

```

def AllUserScores():
    print("All User Scores:")
    try:
        with open(scoreDataFile, "r") as file:
            scores = file.readlines()
            for score in scores:
                userEmail, userScore, date = score.strip().split(",")
                print(f>Email: {userEmail}; Score: {userScore}; Date: {date}")
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")

```

Figure 7: All User Scores Function

Removing Users:

The 'RemoveUser' function removes users from the system based on their email. It deletes the user's data from both the user data file and the score data file by filtering out entries matching the provided email. After successfully removing the user, the action is confirmed. If any file is missing or cannot be opened, it handles the error gracefully by displaying an appropriate message.

```

def RemoveUser():
    print("Remove User:")
    email = input("Enter the email of the user you want to remove: ")
    try:
        with open(userDataFile, "r") as file:
            users = file.readlines()
        with open(userDataFile, "w") as file:
            for user in users:
                storedEmail = user.strip().split(",")[1]
                if storedEmail != email:
                    file.write(user)
        with open(scoreDataFile, "r") as file:
            scores = file.readlines()
        with open(scoreDataFile, "w") as file:
            for score in scores:
                storedEmail = score.strip().split(",")[0]
                if storedEmail != email:
                    file.write(score)
        print(f>User with email '{email}' has been removed.")
    except FileNotFoundError:
        print("An error occurred while trying to open the file. Please try again.")

```

Figure 8: Remove User Function

6. conclusion

This project effectively implements the core features of a Crossword Game, including user authentication, engaging gameplay, score tracking, and administrative functionalities. Modular functions ensure the code is well-organized, making it easy to read, maintain, and extend. The project efficiently manages data without relying on a complex database system by utilizing file-based storage for persistent data. Overall, the project provides a functional and user-friendly crossword puzzle experience while maintaining simplicity in its design and implementation.