**Name: Laiba Adnan**

**Roll no: 004**

**Section: BSDS-2A**

Question 1: Music Streaming Application

In [1]:

**import** random

**class** AudioPlayer:

**def** play(self):

print("Playing audio.")

**def** pause(self):

print("Audio paused.")

**def** stop(self):

print("Audio stopped.")

**class** Playlist:

**def**  init (self): self**.**songs **=** []

**def** add\_song(self, song): self**.**songs**.**append(song)

print(f"'{song}' added to playlist.")

**def** remove\_song(self, song):

**if** song **in** self**.**songs:

self**.**songs**.**remove(song)

print(f"'{song}' removed from playlist.")

**else**:

print(f"'{song}' not found in playlist.")

**def** shuffle(self):

random**.**shuffle(self**.**songs) print("Playlist shuffled.")

**class** StreamingService:

**def**  init (self):

self**.**logged\_in **= False**

self**.**subscribed **= False**

**def** login(self, username): self**.**logged\_in **= True**

print(f"{username} logged in.")

**def** logout(self):

self**.**logged\_in **= False**

print("User logged out.")

**def** check\_subscription(self):

**return** "Active" **if** self**.**subscribed **else** "Inactive"

**class** MusicApp(AudioPlayer, Playlist, StreamingService):

**def**  init (self):

Playlist**.** init (self)

StreamingService**.** init (self)

app **=** MusicApp()

app**.**login("Mahad")

app**.**add\_song("Shape of You") app**.**add\_song("Faded")

app**.**shuffle() app**.**play()

app**.**pause() app**.**stop()

app**.**logout()

Mahad logged in.

'Shape of You' added to playlist. 'Faded' added to playlist.

Playlist shuffled.

Playing audio.

Audio paused.

Audio stopped.

User logged out.

# Question 2: Game Character System

In [3]:

**class** Character:

**def**  init (self, name, level, health): self**.**name **=** name

self**.**level **=** level self**.**health **=** health

**def** move(self):

print(f"{self**.**name} is moving.")

**def** attack(self):

print(f"{self**.**name} attacks!")

**def** level\_up(self): self**.**level **+=** 1

print(f"{self**.**name} leveled up to {self**.**level}.")

**class** Magic:

**def**  init (self): self**.**mana **=** 100 self**.**spells **=** []

**def** cast\_spell(self, spell):

**if** spell **in** self**.**spells **and** self**.**mana **>=** 10: self**.**mana **-=** 10

print(f"Casting {spell}! Remaining mana: {self**.**mana}")

**else**:

print("Not enough mana or unknown spell.")

**def** learn\_spell(self, spell): self**.**spells**.**append(spell)

print(f"Learned new spell: {spell}")

**def** regenerate\_mana(self): self**.**mana **+=** 20

print(f"Mana regenerated to {self**.**mana}")

**class** Wizard(Character, Magic):

**def**  init (self, name, level, health):

Character**.** init (self, name, level, health) Magic**.** init (self)

**def** special\_attack(self):

**if** self**.**health **>** 10 **and** self**.**mana **>=** 10: self**.**health **-=** 10

self**.**mana **-=** 10

print(f"{self**.**name} performs a combo attack! Health: {self**.**health}, Man

**else**:

print("Not enough resources for special attack.")

wiz **=** Wizard("", 1, 100)

wiz**.**learn\_spell("Fireball") wiz**.**move()

wiz**.**cast\_spell("Fireball") wiz**.**attack()

wiz**.**special\_attack()

Learned new spell: Fireball is moving.

Casting Fireball! Remaining mana: 90 attacks!

performs a combo attack! Health: 90, Mana: 80

# Question 3: Document Editor Components

In [4]:

**class** TextEditor:

**def** add\_text(self, text):

print(f"Text added: {text}")

**def** delete\_text(self):

print("Text deleted.")

**def** select\_text(self):

print("Text selected.")

**class** SpellChecker:

**def** check\_spelling(self):

print("Spelling checked.")

**def** suggest\_corrections(self):

print("Suggestions provided.")

**class** Formatter:

**def** change\_font(self):

print("Font changed.")

**def** change\_style(self):

print("Style changed.")

**def** change\_alignment(self):

print("Alignment changed.")

**class** DocumentEditor(TextEditor, SpellChecker, Formatter):

**def** save\_document(self, format):

print(f"Document saved as {format}.")

editor **=** DocumentEditor()

editor**.**add\_text("Hello World") editor**.**check\_spelling()

editor**.**change\_style()

editor**.**save\_document("PDF")

Text added: Hello World Spelling checked.

Style changed.

Document saved as PDF.

# Social Media Platform

In [ ]:

**class** User:

**def**  init (self, name): self**.**name **=** name

self**.**friends **=** []

**def** add\_friend(self, friend): self**.**friends**.**append(friend)

print(f"{friend} added as friend.")

**class** ContentCreator:

**def** create\_post(self, content):

print(f"Post created: {content}")

**def** delete\_post(self):

print("Post deleted.")

**def** edit\_post(self, new\_content):

print(f"Post updated to: {new\_content}")

**class** Analytics:

**def**  init (self): self**.**views **=** 0

self**.**likes **=** 0

**def** track\_views(self, count): self**.**views **+=** count

**def** track\_likes(self, count): self**.**likes **+=** count

**def** generate\_report(self):

print(f"Views: {self**.**views}, Likes: {self**.**likes}")

**class** Influencer(User, ContentCreator, Analytics):

**def**  init (self, name):

User**.** init (self, name) Analytics**.** init (self)

**def** calculate\_earnings(self):

earnings **=** self**.**views **\*** 0.01 **+** self**.**likes **\*** 0.05 print(f"Earnings: ${earnings:.2f}")

influencer **=** Influencer("Maharaj Zeel") influencer**.**add\_friend("Ali")

influencer**.**create\_post("New Vlog!") influencer**.**track\_views(1000)

influencer**.**track\_likes(300)

influencer**.**calculate\_earnings()

Ali added as friend.

Post created: New Vlog! Earnings: $25.00

# Smart Home System

In [9]:

**class** Device:

**def** power\_on(self):

print("Device powered on.")

**def** power\_off(self):

print("Device powered off.")

**def** status(self):

print("Device status: OK")

**class** Sensor:

**def** read\_temperature(self):

**return** 26

**def** read\_humidity(self):

**return** 60

**def** read\_motion(self):

**return True**

**class** NetworkNode:

**def** connect(self):

print("Connected to network.")

**def** disconnect(self):

print("Disconnected from network.")

**def** send\_data(self, data):

print(f"Data sent: {data}")

**class** SmartHomeController(Device, Sensor, NetworkNode):

**def** automate(self):

temp **=** self**.**read\_temperature() motion **=** self**.**read\_motion()

**if** temp **>** 25 **and** motion: self**.**power\_on()

self**.**send\_data("Cooling system activated.")

home **=** SmartHomeController() home**.**connect()

home**.**automate() home**.**status()

home**.**disconnect()

Connected to network.

Device powered on.

Data sent: Cooling system activated. Device status: OK

Disconnected from network.