AI 프로젝트 기술 보고서

프로젝트명	AI 면접 훈련도구 (AI-based mock interview training)	팀명	마이너 클래스

1. 프로젝트 개요

1.1. 제작 배경 및 동기

최근 대기업에서 AI를 통한 역량 검사 및 비대면 면접을 채택하는 비중이 증가하고 있으며, 이러한 추세는 향후 산업 전반으로 확산될 것이라 예측됩니다. AI 면접은 기존의 대면 면접의 면접관이 가지는 주관적인 가치관으로 일어날 수 있는 면접관 편향 문제를 감소시키는 효과가 있으나, AI 시스템 자체가 완벽하게 공정한 평가자의 역할을 수행하는 데에는 한계가 있다는 문제의식으로 본 프로젝트가 시작되었습니다.

구체적으로 AI가 '정상적이거나 선호되는' 비언어적 소통 패턴을 보이는 지원자를 선별하도록 설계된 경우, 이 과정에서 또 다른 형태의 암묵적 편향이 발생할 수 있습니다. 이에 본 프로젝트에서는 면접 상황에서 겪는 비언어적 소통의 어려움을 해소하기 위해 AI 기반 실시간 피드백 훈련 도구를 개발하기로 결정하였습니다.

이 보조 도구의 최종 목표는 AI의 객관적인 측정 능력을 활용하여, 신체 움직임(예: 몸을 흔드는 행위)이나 시선 처리(예: 눈 맞춤의 어려움) 등의 비언어적 요소에서 어려움을 겪는 '신경 다양인'과 같은 소수자들에게 자신의 면접 태도에서 문제가 될 수 있는 요소를 객관적으로 인지하고 개선할 수 있도록 돕는 데 있습니다.

1.2 AI 비전과의 연계

본 프로젝트는 멀티모달 AI 분석 및 데이터 퓨전(Data Fusion) 기술을 통해 객관적인 역량평가 환경을 구축하는 걸 목표로 하고 있습니다. 기존의 정성적 평가 방식에서 벗어나, 면접 영상으로부터 추출된 비언어적 데이터를 정량적으로 수치화하고 이를 통합 분석하는 것이 핵심입니다.

- **AI 비전을 활용한 시각 데이터 분석**
  - ❍ 시선 추적 모델(OpenCV Gaze Tracking)을 통한 시선처리 방향 데이터 실시간 수집
  - ❍ 자세 추적 모델(MediaPipe Posture measurement)을 통한 포즈 데이터 실시간 수집
    - Pose Landmark Detection
    - skeleton 탐지를 통해 관절의 위치변화량으로 움직임 감지
    - skeleton 데이터 중 귀와 코 사이의 거리를 통해 대상과의 거리를 추정하여 수치 보정
- **DL 모델을 통한 음성 데이터 분석**
  - ❍ 마이크로 입력된 음성 데이터를 STT(Speech-to-Text)로 변환하여 면접 답변의 내용 분석
  - ❍ 음성 떨림 정도를 측정하여 지원자의 불안감 및 긴장도를 객관적으로 파악
- **LLM 모델 Gemini를 활용한 피드백 리포트 생성**
  - ❍ 태도 점수(시선처리, 자세 불안정성)와 3가지의 역량평가 게임 점수를 결합하여, Gemini api를 통해 프롬포트로 전달하여 피드백 리포트 생성
  - ❍ 시각, 음성, 정량 데이터를 융합하여 입체적인 피드백 리포트 제공을 목표

2. 개발 조직 및 환경

2.1. 팀원 구성 및 역할 분담

성명	담당 역할	주요 기여 내용
민해인	PM	- 프로젝트를 총괄하고 버전 현상 관리 - 프로젝트 이슈, 일정 관리
서준희	UI	- UI design, 사용자 인터페이스를 정의하고 구현 - 시연 영상 제작
이승현	Lead Programmer	- AI modeling. AI model 선택. Data 수집. Training 수행 - 응답 분석 알고리즘 설계 및 구현
이준혁	Design & Game build	- UI design, 사용자 인터페이스를 정의하고 구현한다. - 역량 게임 로직을 구현한다.

2.2. 개발 환경 및 기술 스택

■ **개발환경**

- ❍ 운영체제(OS) : Linux Ubuntu 22.04 LTS
- ❍ 버전 관리 : Git / Github
- ❍ UI/UX 설계 : Figma (화면 구성 및 디자인 프토로타이핑)

■ **AI/LLM**

- ❍ 시선추적 : OpenCV-python (Gaze Tracking 모듈)
- ❍ 자세추적 : MediaPipe (Posture measurement 모듈)
- ❍ LLM API : 0Google Gemini 2.5 API (최종 피드백 리포트 생성)

■ **Backend 및 로직 처리**

- ❍ 개발언어 : Python3.x
- ❍ 이미지 처리 : Pillow(PIL) 라이브러리 (이미지 투명화 및 가공)

■ **Frontend (GUI 환경)**

- ❍ GUI 프레임워크 : Tkinter (사용자 인터페이스 구축 및 최종 화면 출력)

3. 주요 개발 내용 및 구현 상세

3.1. 시스템 아키텍처

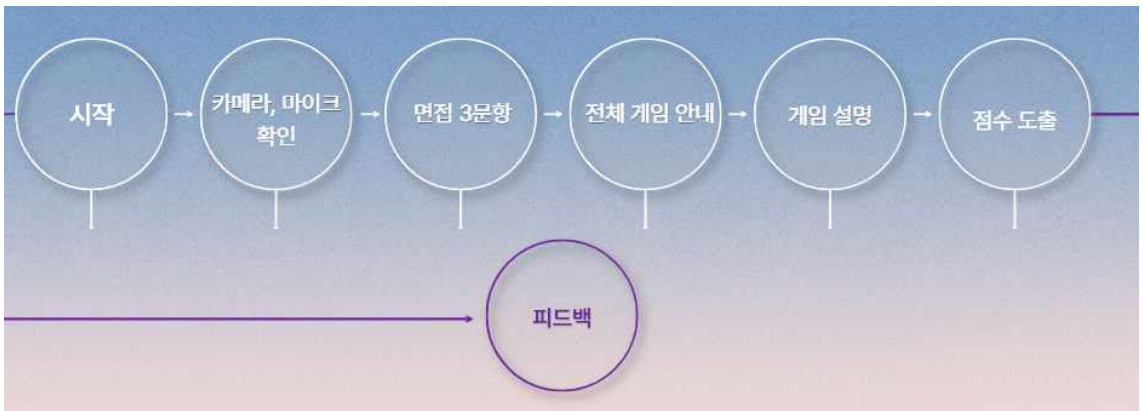


그림 1 진행 프로세스

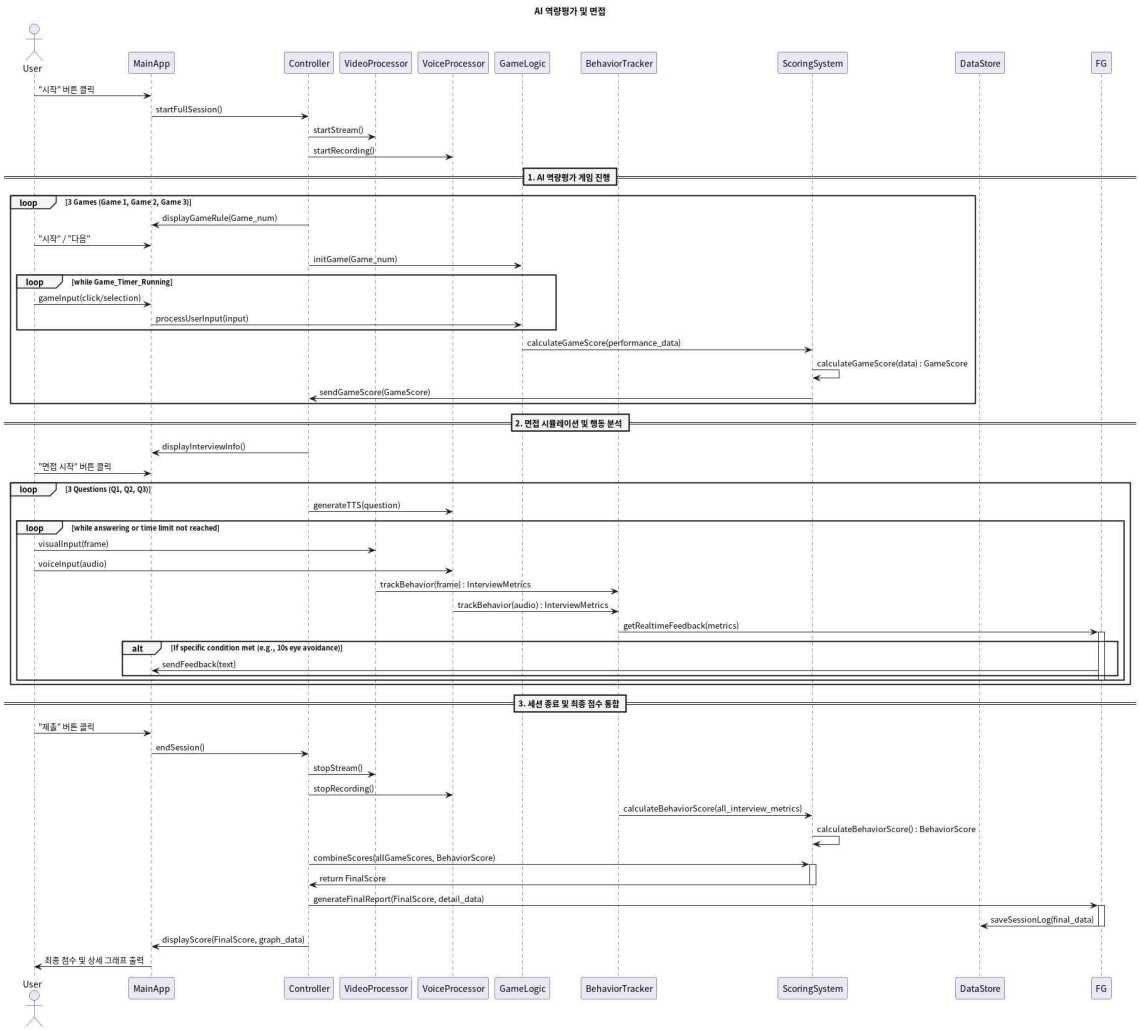


그림 2 시퀀스 다이어그램 UML

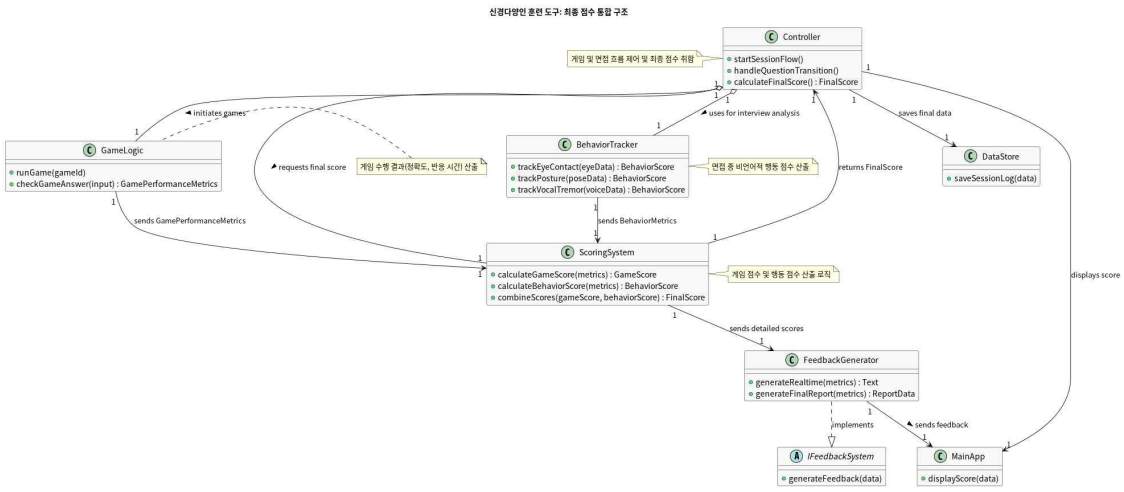


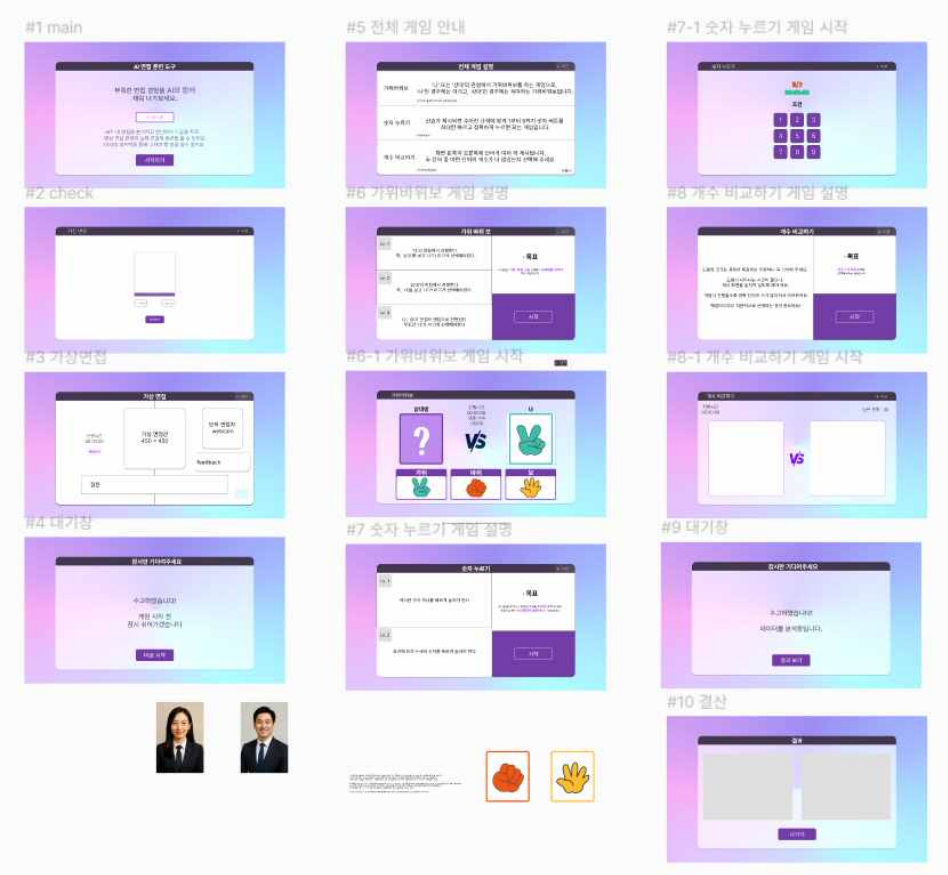
그림 3 클래스 다이어그램 UML

3.2. 핵심 기능별 구현 상세

3.2.1 사용자 인터페이스(UI/UX)

■ **Figma를 활용한 인터페이스 디자인**

❍ 인터페이스 구성



3.2.2 면접 로직

■ **대기화면**

❍ 카메라와 마이크가 연결될 시 '준비 완료' 버튼 활성화

■ **면접 화면**

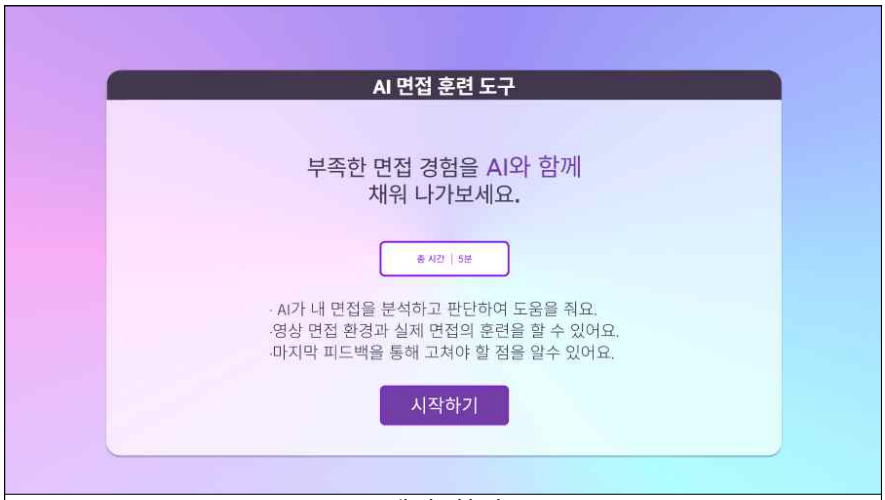
❍ 면접 질문

- 제미나이 API를 통해 받아온 면접 질문 데이터를 질문 영역에 표시
- API를 통해 데이터를 받아오는 데 걸리는 시간이 약 10초
- 그동안 애플리케이션 동작이 멈추는 오류 존재
- 성능 향상을 위해 리스트에 예상 질문을 넣고 순차적으로 표시하는 방법으로 대체

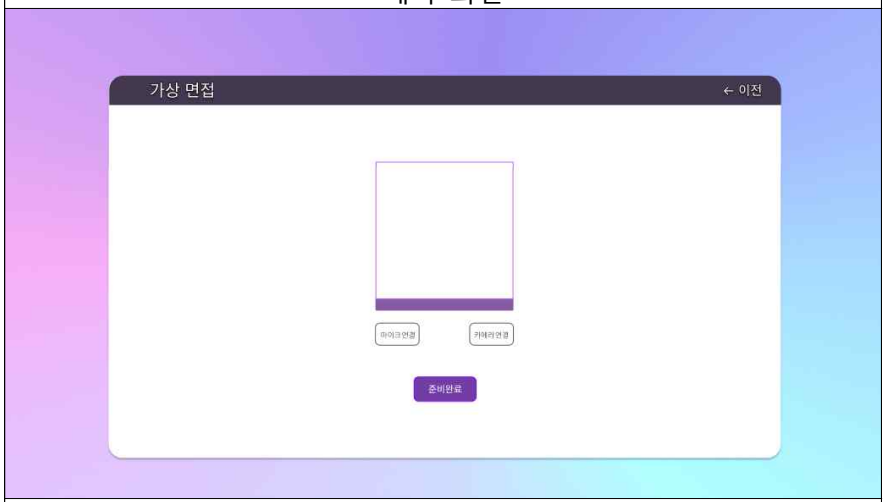
❍ 시선 처리와 자세 불안정에 관한 피드백 표시

- 캠으로 들어온 영상을 스트리밍으로 복사하여 AI 비전 판독
- Gaze Tracking 모델로 시선을 추적. 시선의 위치가 center가 아닌 시간을 측정
- Posture measurement 모델을 통한 자세 추적. Skeleton의 위치 변화가 존재하는 시간을 측정.
- 눈 사이의 거리로 카메라와 사람의 거리를 판단. 이를 통해 판단 민감성을 조정

■ 구현 **화면**



대기 화면



카메라 세팅 및 준비화면

면접 화면

3.2.3 역량평가 게임 로직

■ **가위바위보 게임**

❍ 게임 화면 – 가위바위보 게임

가위바위보 게임 화면

❍ 게임 로직

- 컴퓨터의 선택에 True/False를 값을 랜덤으로 받고, '가위', '바위', '보' 랜덤 값을 컴퓨터의 선택이 True일 때는 상대방 영역에 False일 때는 자신의 영역에 이미지가 표시
- Winner 판단 딕셔너리를 생성. 딕셔너리를 기준으로 정답 여부를 판정.
- 푼 문제 대비 맞춘 개수로 점수 추출

❍ 실제 코드

```
def prepare_new_round(self):
    self.computer_is_actor = random.choice([True, False])
    self.img_card_pick = random.choice(CHOICES)
    if self.computer_is_actor:
        self.opponent_choice = self.choice_images[self.img_card_pick]
        self.my_choice = self.choice_images["blank"]
    else:
        self.opponent_choice = self.choice_images["blank"]
        self.my_choice = self.choice_images[self.img_card_pick]
    self.update_ui()

def play_click(self, clicked_choice: str):
    if not self.is_game_running:
        return

    self.total_tries += 1

    if self.computer_is_actor:
        opponent_card = self.img_card_pick
```

```
            my_card = clicked_choice
            self.opponent_choice = self.choice_images[opponent_card]
            self.my_choice = self.choice_images[my_card]
        else:
            my_card = self.img_card_pick
            opponent_card = clicked_choice
            self.opponent_choice = self.choice_images[opponent_card]
            self.my_choice = self.choice_images[my_card]

        if WIN_RULE[my_card] == opponent_card:
            self.correct_count += 1

        self.update_ui()
        self.after(200, self.prepare_new_round)
```

■ **숫자 누르기 게임**

❍ 게임 화면 – 숫자 누르기 게임

숫자 누르기 게임

❍ 게임 로직

- 1~9까지의 숫자를 기본으로 포함. 숫자 키패드에 무작위로 숫자 뿌리기
- 30%의 확률로 각 숫자를 2~4회 반복하여 정답 조건을 무작위로 생성
- 시간당 푼 개수로 점수 추출

❍ 실제 코드

```
# 시퀀스 생성 (generate_sequence)
def generate_sequence(self):
    seq = []
    for num in range(1, 10):
        repeat = 1
```

```
        if random.random() < 0.3: # 30% 확률로 반복
            repeat = random.randint(2, 4)
        seq.extend([num] * repeat) # 숫자를 반복 횟수만큼 추가
    return seq

# 조건 설명 (sequence_description)
def sequence_description(self, seq):
    counts = {}
    for num in seq:
        counts[num] = counts.get(num, 0) + 1
    # 2회 이상 반복되는 숫자에 대한 규칙만 생성
    desc_list = [f"숫자 {num}는 {counts[num]}번 누르세요."
                 for num in range(1, 10) if num in counts and counts[num] > 1]
    # 규칙이 없으면 기본 문구 반환
    base = " ".join(desc_list) if desc_list else "1~9까지 순서대로 누르세요."

    # 텍스트 길이에 맞춰 줄바꿈 처리
    max_len = 46
    lines = [base[i:i+max_len] for i in range(0, len(base), max_len)]
    return "\n".join(lines)
```

■ **개수 비교하기 게임**

❍ 게임 화면 – 숫자 누르기 게임

개수 비교하기 게임

❍ 게임로직

- 작성해 둔 단어 리스트에서 랜덤하게 2종류를 추출.
- 각 영역의 개수가 같지 않도록 랜덤하게 숫자 추출
- 개수가 더 많은 숫자를 선택할 때 정답처리

○ 실제 코드

```
    # 리소스 / 설정
    self.WORDS = ["준혁", "해인", "승현", "준희","Intel","마스터","마이너","클래스","화이팅"]
    self.TOTAL_QUESTIONS = 10
    self.LEFT_AREA = (336, 388, 836, 888)
    self.RIGHT_AREA = (1085, 388, 1585, 888)

def bind_events(self):
    # 캔버스 클릭 핸들링
    self.canvas.bind("<Button-1>", self.handle_canvas_click)

def handle_canvas_click(self, event):
    # 정답 제출 허용 상태에서만 클릭 처리
    if not self.game_running or not self.answer_allowed:
        return

    x, y = event.x, event.y
    if self.LEFT_AREA[0] <= x <= self.LEFT_AREA[2] and self.LEFT_AREA[1] <= y <=
self.LEFT_AREA[3]:
        self.judge_answer(self.current_left_word)
    elif self.RIGHT_AREA[0] <= x <= self.RIGHT_AREA[2] and self.RIGHT_AREA[1] <=
y <= self.RIGHT_AREA[3]:
        self.judge_answer(self.current_right_word)

def update_timer(self):
    if not self.game_running or self.start_time is None:
        return

    elapsed = int(time.time() - self.start_time)
    h, m, s = elapsed // 3600, (elapsed % 3600) // 60, elapsed % 60
    self.canvas.itemconfig(self.timer_text_id, text=f"진행시간\n{m:02d}:{s:02d}")

    if self.timer_after_id:
        self.after_cancel(self.timer_after_id)

    self.timer_after_id = self.after(1000, self.update_timer)

def show_words_once(self):
    # 이전 word 태그 삭제 및 안내 초기화
    self.canvas.delete("word")
```

```
        self.info_label.config(text="")

        # 랜덤 단어/카운트 선정 (같은 단어가 양쪽에 오지 않도록 처리)
        self.current_left_word = random.choice(self.WORDS)
        right_candidates = [w for w in self.WORDS if w != self.current_left_word]
        self.current_right_word = random.choice(right_candidates)

        # 각 영역의 개수 결정 (같지 않도록)
        self.count_left = random.randint(self.MIN_WORDS, self.MAX_WORDS)
        self.count_right = random.randint(self.MIN_WORDS, self.MAX_WORDS)
        while self.count_right == self.count_left:
            self.count_right = random.randint(self.MIN_WORDS, self.MAX_WORDS)

        # 정답은 개수가 더 많은 쪽의 단어
        self.answer_expected = self.current_left_word if self.count_left > self.count_right
else self.current_right_word

        # 화면에 단어들 배치 (tag="word")
        self.place_word(self.LEFT_AREA, self.current_left_word, self.count_left, "word")
        self.place_word(self.RIGHT_AREA, self.current_right_word, self.count_right, "word")

        # 단어 노출 후 선택 가능하도록 ask_answer 호출 (2초 후)
        self.answer_allowed = False
        self.after(2000, self.ask_answer)
```

3.2.3 응답 분석 및 면접 평가 로직

■ **JEMINI API를 활용한 피드백 리포트 생성**

❍ jemini api를 통해 위에서 추출한 점수들도 최종 피드백 리포트 생성

❍ 프롬프트

```
prompt = "가위,바위,보 게임은 나또는 상대의 관점에서 가위바위보를 하는 게임으로, " \
         "나인 경우에는 이기고, 상대인 경우에는 져야하는 게임이다." \
         "숫자 누르기 게임은 신호가 제시되면 주어진 규칙에 맞게 1부터 9까지 " \
         "숫자 버튼을 최대한 빠르고 정확하게 누르면 되는 게임입니다. " \
         "개수 비교하기 게임은 화면 왼쪽과 오른쪽에 단어가 여러 개 제시됩니다." \
         "두 단어 중 어떤 단어의 개수가 더 많았는지 선택하는 게임야." \
         "이 게임들의 점수를 가지고 각 게임들에게 필요한 요소, 능력, 역량, " \
         "요구사항, 기술 등을 키워드로 삼아 개선점이나 보완해야 할 역량들을 " \
         "서술해줘. " \
         "50자 이내로그리고 집중 안 한시간, 떨어버린 시간이 있는데 그것은" \
         "면접 중 집중 안 한 시간은 시선 처리, 떨어버린 시간은 몸의 떨림을 통해" \
```

```
            "도출해낸 점수야. " \
            "이것을 통해 어땠는지 그리고 다음 향후 방안이나 개선점을 통합점으로 " \
            "설명해줘. " \
            "50자 이내로 그리고 마지막으로 이 모든 것을 통합하여 심리 상태에 대해 " \
            "분석하고, 개선 방안을 포괄적으로 만들어줘 100자 이내로" + self.text
```

4. 개발 결과 및 성과 (Deliverables and Results)

4.1. 최종 결과물 (Deliverables)

주요 산출물: https://github.com/MinorClass/AI_Project.git

5. 결론 및 향후 계획 (Conclusion and Future Plan)

5.1. 결론

본 프로젝트에서 제작한 면접 훈련 도구는 단지 일회성 보조 수단이 아니라, 기업과 사회가 다양한 사고방식의 가치를 인정하고 투자하고 있음을 보여주는 포용적 고용의 핵심 인프라가 되어야 합니다. 성공적인 도구 개발은 면접관의 잠재적인 편향을 제거하고 더 다양한 생각을 가진 사람들의 노동 시장 참여율을 높이는 사회적, 경제적 효과를 가져올 것입니다.

5.2. 향후 개선 및 확장 방향

■ **서비스 고도화 방안**

❍ AI 기반 비언어/언어 데이터 통합 분석 기능 도입
- Tkinter의 환경적 제약에서 벗어난 GUI 환경으로 전환.
- 카메라 및 마이크 동시 접근성을 확보하여 표정, 시선, 음성 데이터를 통합 수집.

❍ Gemini API 연동을 통한 실제 면접관 수준의 대화 및 압박 면접 시뮬레이션 구현
- 응답자의 SST(Speech-to-Text) 변환 후, Gemini API로 맥락 인지형 면접 대화를 진행하여, 답변의 논리성 및 구조에 대한 심층 피드백 제공

❍ 무의식적 행동 분석 및 피드백
- 게임 진행 중 발생하는 마우스 클릭 빈도, 움직임 패턴 등의 미세한 행동 데이터를 측정하여, 면접 시의 불안정서 지표로 활용하여 안정화 훈련 피드백 제공.

■ **개선 과제 및 기술적 한계 극복**

❍ Tkinter 환경에서 마이크 및 카메라의 동시 접근이 원활하지 않아 기능 구현 불가
- Python 기반 GUI 프레임워크 또는 웹 프레임워크 기반으로 전환하여 Hardware 접근성 문제를 근본적으로 해결

❍ SST 미구현으로 Gemini API 활용을 한 실제 대화형 면접이 불가능했으며, 음성 떨림 분석 등 감정 분석 기능 구현 실패
- Google Speech0to-Text API 또는 유사 라이브러리를 도입하여, AI 분석 모델에 Gemini의 맥락 유지 및 상황 판단 능력을 결합하여 대화의 품질을 극대화

❍ 마우스 클릭 빈도 등 무의식적 행동 데이터 측정 및 피드백 기능 미구현
- 클릭 이벤트의 횟수 및 속도를 정량적으로 기록하는 기술을 우선 개발