Meeting Minutes Assistant

1 Project Background and Objectives

1.1 Project Background

In modern work environments, meetings are a crucial part of team collaboration, decision-making, and information sharing. However, the task of compiling meeting minutes after a meeting is often tedious and time-consuming. Traditional manual note-taking methods are not only inefficient but also prone to missing critical information, which can lead to miscommunication or redundant discussions in follow-up work. With the rapid advancement of artificial intelligence (AI) technologies, particularly in the fields of speech recognition and natural language processing (NLP), it has become possible to automate the generation of meeting minutes using AI.

This project aims to develop an efficient and intelligent Meeting Minutes Assistant by leveraging state-of-the-art speech recognition models (Whisper-large-v3) and powerful text generation models (ChatGPT). The tool is designed to quickly convert meeting recordings into structured meeting minutes, thereby improving work efficiency and reducing the need for manual intervention.

1.2 Social Pain Points

- **High Time Cost**: Manually compiling meeting minutes can take hours, especially for lengthy meetings, significantly draining productivity.
- **Information Loss**: Human note-takers often miss important details, particularly during complex or fast-paced discussions.
- **Inconsistent Quality**: The quality of meeting minutes varies depending on the note-taker’s style and focus, leading to inconsistencies.
- **Language Barriers**: In multinational or multilingual teams, language differences can further complicate the process of recording and summarizing meetings.

1.3 Target Audience

- **Corporate Teams**: Companies or organizations that require efficient recording and sharing of meeting content, especially those with frequent meetings.
- **Educational Institutions**: Used for documenting academic discussions, seminars, or course content.
- **Freelancers**: Professionals who need to record communications with clients or partners for follow-up purposes.

- **Multilingual Teams**: Teams that require support for multiple languages to overcome language barriers in meeting documentation.

1.4 Project Objectives

- **Efficiency**: Reduce the time required to convert meeting recordings into minutes from hours to just a few minutes, significantly enhancing productivity.
- **Accuracy**: Utilize advanced speech recognition and text generation technologies to ensure the completeness and accuracy of meeting minutes, minimizing information loss.
- **Structured Output**: Generate well-organized and easy-to-read meeting minutes, including key elements such as meeting topics, discussion points, decisions made, and action items.
- **Multilingual Support**: Enable the generation of meeting minutes in multiple languages to cater to the needs of international teams.
- **User-Friendliness**: Provide a simple and intuitive interface with a seamless workflow, lowering the barrier to entry and improving user experience.

By achieving these objectives, this project aims to deliver a highly efficient and intelligent Meeting Minutes Assistant that empowers users to focus on more valuable tasks, freeing them from the burdens of manual note-taking.

2 Related Technology Survey

2.1 Speech-to-Text Technologies

Speech-to-text (STT) technology is a critical component of this project, as it converts spoken language from meeting recordings into written text. Several advanced models and tools are available for this purpose:

- **Whisper Model**:
  - Developed by OpenAI, Whisper is a state-of-the-art automatic speech recognition (ASR) system trained on a large and diverse dataset.
  - The whisper-large-v3 variant offers high accuracy and supports multiple languages, making it ideal for multilingual meeting environments.
  - Its ability to handle noisy audio and varying accents makes it a robust choice for real-world applications.
- **Google Speech-to-Text**:
  - A cloud-based service that provides accurate transcription with support for over 120 languages.
  - Offers features like automatic punctuation, speaker diarization, and noise cancellation.
  - However, it requires an internet connection and may incur higher costs for large-scale usage.

- **Microsoft Azure Speech Service**:
  - Another cloud-based solution that supports real-time and batch transcription.
  - Includes features like custom language models and speech translation.
  - Similar to Google's service, it relies on cloud infrastructure and may raise privacy concerns for sensitive data.
- **Comparison**:
  - Whisper stands out for its open-source nature, offline capabilities, and multilingual support, making it a preferred choice for this project.
  - Cloud-based solutions like Google and Azure offer additional features but are less suitable for scenarios requiring local deployment and data privacy.

2.2 Natural Language Processing (NLP) Technologies

NLP technologies are used to process and summarize the transcribed text into structured meeting minutes. Key models and methods include:

- **ChatGPT (OpenAI)**:
  - A powerful language model based on the GPT-4 architecture, capable of generating coherent and contextually relevant text.
  - It excels in summarization tasks, making it ideal for condensing meeting transcripts into concise and actionable minutes.
  - Its ability to follow instructions via prompts allows for customization of the output format and content.
- **BERT (Bidirectional Encoder Representations from Transformers)**:
  - A pre-trained model designed for understanding context in text.
  - While BERT is highly effective for tasks like sentiment analysis and question answering, it is less suited for generative tasks like summarization.
- **T5 (Text-to-Text Transfer Transformer)**:
  - A versatile model that frames all NLP tasks as a text-to-text problem.
  - It performs well on summarization tasks but requires fine-tuning for specific use cases, which can be resource-intensive.
- **Comparison**:
  - ChatGPT is chosen for this project due to its superior generative capabilities, ease of use, and ability to handle complex instructions through prompt engineering.
  - While BERT and T5 are strong alternatives, they are less flexible for generating structured and contextually rich meeting minutes.

2.3 Prompt Engineering

Prompt engineering is a crucial aspect of leveraging ChatGPT for meeting minutes generation. It involves designing effective prompts to guide the model in producing the desired output:

Key Considerations:

- Clarity: Prompts must be clear and specific to ensure the model understands the task.
- Context: Providing sufficient context (e.g., meeting type, participants) helps the model generate more relevant summaries.
- Formatting: Instructions on output format (e.g., bullet points, headings) ensure consistency and readability.

  Examples of Prompts:
- "Summarize the following meeting transcript into structured meeting minutes, including key discussion points, decisions made, and action items."
- "Identify the main topics discussed in the meeting and list the responsibilities assigned to each participant."

  Challenges:
- Balancing brevity and completeness in the generated minutes.
- Handling ambiguous or incomplete information in the transcript.

2.4 Integration of Technologies

The integration of Whisper and ChatGPT forms the core of this project:

- Whisper converts the audio input into a text transcript.
- The transcript is preprocessed to remove noise, segment text, and enhance readability.
- ChatGPT processes the cleaned transcript using carefully designed prompts to generate structured meeting minutes.

2.5 Alternative Approaches

- **End-to-End Solutions**: Some tools, like Otter.ai, offer integrated speech-to-text and summarization features. However, they often lack customization and may not support local deployment.
- **Rule-Based Systems**: Traditional methods rely on predefined templates and rules for summarization. While simple, they lack the flexibility and adaptability of AI-driven approaches.

3 Project Structure and Functionality

The Meeting Minutes Assistant is designed as a modular and scalable system, leveraging Python and PyTorch for local deployment of the Whisper-large-v3 model and integrating OpenAI's ChatGPT API for text summarization. Below is a detailed breakdown of the project structure and the functionality of each component.

3.1 `WhisperASR.py`

This module handles the automatic speech recognition (ASR) task using the Whisper-large-v3 model.

- **Key Features**:
  - **Initialization**: Configures the Whisper model with parameters such as device (CPU/GPU), language, task (transcribe/translate), and chunk length.
  - **Audio Processing**: Accepts audio files (single or batch) and converts them into text transcripts.
  - **Timestamp Support**: Optionally provides word-level timestamps for detailed analysis.
  - **Efficiency**: Optimized for low CPU memory usage and supports batch processing for faster transcription.
- **Workflow**:
  - Loads the Whisper model and processor.
  - Converts audio files into a dataset format compatible with the model.
  - Processes the audio in chunks and generates text transcripts.

```
class WhisperASR:
    def __init__(self,
                 device=None,
                 max_new_tokens=128,
                 language='english',
                 task='transcribe',
                 chunk_length_s=30,
                 batch_size=16,
                 word_level_timestamps=False):
        self.device = device if device is not None else 'cuda:0' if torch.cuda.is_available() e
        torch_dtype = torch.float16 if torch.cuda.is_available() else torch.float32
        model_id = 'openai/whisper-large-v3'
        model = AutoModelForSpeechSeq2Seq.from_pretrained(model_id,
                                                          torch_dtype=torch_dtype,
                                                          low_cpu_mem_usage=True,
                                                          use_safetensors=True)
        model.to(device)
        processor = AutoProcessor.from_pretrained(model_id)
        self.pipe = pipeline('automatic-speech-recognition',
                             model=model,
                             tokenizer=processor.tokenizer,
                             feature_extractor=processor.feature_extractor,
                             chunk_length_s=chunk_length_s,
                             max_new_tokens=max_new_tokens,
                             batch_size=batch_size,
                             return_timestamps=True if not word_level_timestamps else 'word',
                             torch_dtype=torch_dtype,
                             device=self.device)
        self._kwargs = {'language': language, 'task': task}

    def __call__(self, audio_file: Union[str, Iterable]) -> Union[str, Iterable]:
        ds = Dataset.from_dict({'audio': [audio_file] if isinstance(audio_file, str) else audio
            'audio', Audio(sampling_rate=16000))
        if isinstance(audio_file, str):
            return self._handle(ds)[0]
        else:
            return self._handle(ds)

    def _handle(self, ds: Dataset) -> list:
        res = []
        for data in ds:
            sample = data['audio']
            pred = self.pipe(sample.copy(), generate_kwargs=self._kwargs)
            res.append(pred['chunks'])
        return res
```

3.2 `ChatGPT.py`

This module interacts with OpenAI's ChatGPT API to generate summaries and structured meeting minutes.

- **Key Features**:
  - **Initialization**: Configures the ChatGPT model with an API key and model version (e.g., `gpt-3.5-turbo-16k`).
  - **Request Handling**: Sends text prompts to the ChatGPT API and retrieves the generated responses.
  - **Error Handling**: Implements retry logic for handling API connection errors or rate limits.
- **Workflow**:
  - Constructs a prompt with a system role (e.g., "Summarize the meeting transcript") and user input (the transcript).
  - Sends the prompt to the ChatGPT API and returns the generated text.

```
class ChatGPT:
    def __init__(self, api_key: str, model='gpt-3.5-turbo-16k', try_times=5) -> None:
        self.client = OpenAI(api_key=api_key)
        self.model = model
        self.try_times = try_times

    def request(self, text: str, role: str):
        for times in range(self.try_times):
            try:
                completions = self.client.chat.completions.create(model=self.model, messages=[
                    {'role': 'system', 'content': role},
                    {'role': 'user', 'content': text}
                ])
                return completions.choices[0].message.content
            except RateLimitError or APIConnectionError as e:
                print(f'WARNNING: connect openAI error. reason: {e}\nWe will try it again.')
                time.sleep((times + 1) * 5)
            if times == self.try_times - 1:
                raise OpenAIConnectError(f'We have tried to connect to chatGPT API {self.try_t
```

3.3 `Role.py`

This module defines the roles and prompts used for generating meeting minutes.

- **Key Features**:
    - **Role Initialization**: Loads predefined prompts from markdown files for specific tasks (e.g., summarizing, editing).
    - **Prompt Customization**: Allows dynamic replacement of placeholders in prompts with user-defined templates.
- **Workflow**:
    - Reads the role-specific prompt from a file.
    - Sends the prompt and input text to the ChatGPT API for processing.

```python
class Role:
    def __init__(self, role_file: str, model: ChatGPT) -> None:
        self.model = model
        self.role_prompt = self._read_role_prompt(role_file)

    def _read_role_prompt(self, role_file):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/{role_file}', 'r', encoding='utf-8') as f:
            return f.read()

    def request(self, text: str):
        return self.model.request(text, self.role_prompt)
```

3.4 `RoleList.py`

This module defines specific roles for different tasks, such as generating meeting minutes, writing summaries, and editing content.

- **Key Roles**:
    - **MeetingSecretary**: Generates structured meeting minutes using a predefined template.
    - **SummaryWriter**: Summarizes long transcripts into concise segments.
    - **MeetingMinutesEditor**: Reviews and refines the generated minutes for accuracy and completeness.
- **Workflow**:
    - Initializes the role with a ChatGPT instance and a template.
    - Processes the input text and generates the desired output based on the role’s prompt.

```
class MeetingSecretary(Role):
    def __init__(self, model, template: str = None) -> None:
        super().__init__('MeetingSecretary.md', model)
        self.raw_prompt = self.role_prompt
        self.set_template(template if template is not None else self._get_default_template())

    def set_template(self, template: str) -> str:
        self.role_prompt = self.raw_prompt.replace('${template}', template)

    def _get_default_template(self):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/DefaultMinutesTemplate.md', 'r', encoding='utf-8') as f:
            return f.read()

class SummaryWriter(Role):
    def __init__(self, model: ChatGPT) -> None:
        super().__init__('SummaryWriter.md', model)

class MeetingMinutesEditor(Role):
    def __init__(self, model, template: str = None) -> None:
        super().__init__('MeetingMinutesEditor.md', model)
        self.raw_prompt = self.role_prompt
        self.set_template(template if template is not None else self._get_default_template())

    def set_template(self, template: str) -> str:
        self.role_prompt = self.raw_prompt.replace('${template}', template)

    def _get_default_template(self):
        path = os.path.dirname(os.path.abspath(__file__))
        with open(f'{path}/prompts/DefaultMinutesTemplate.md', 'r', encoding='utf-8') as f:
            return f.read()
```

3.5 `main.py`

This is the entry point of the application, orchestrating the workflow from audio input to final meeting minutes.

- **Key Features**:
  - **Argument Parsing**: Accepts user inputs such as audio file path, output file name, language, and device configuration.
  - **ASR Execution**: Calls the WhisperASR module to transcribe the audio file.
  - **Summary Generation**: Divides the transcript into segments and summarizes each segment using the SummaryWriter role.

    - **Minutes Generation**: Uses the MeetingSecretary role to generate structured meeting minutes.
    - **Output Validation**: Reviews the final output using the MeetingMinutesEditor role and saves it to a file.
- **Workflow**:
    i. Parse command-line arguments.
    ii. Transcribe the audio file using WhisperASR.
    iii. Summarize the transcript using the SummaryWriter role.
    iv. Generate meeting minutes using the MeetingSecretary role.
    v. Validate and save the output using the MeetingMinutesEditor role.

3.6 Modular Design and Scalability

The project is designed with modularity in mind, allowing for easy integration of new features or replacement of components. For example:

- The Whisper model can be replaced with other ASR systems.
- The ChatGPT API can be swapped with alternative text generation models.
- New roles can be added to support additional tasks, such as sentiment analysis or action item extraction.

3.7 Functionality Overview

1. **Audio Transcription**: Converts meeting recordings into text using Whisper-large-v3.
2. **Text Summarization**: Breaks down long transcripts into manageable segments and summarizes them.
3. **Minutes Generation**: Produces structured meeting minutes using predefined templates.
4. **Output Validation**: Ensures the accuracy and completeness of the generated minutes.
5. **User Customization**: Allows users to specify templates, languages, and output formats.

4 Performance Evaluation

To ensure the efficiency and accuracy of the Meeting Minutes Assistant in real-world applications, we conducted a comprehensive performance evaluation. The evaluation focuses on the following aspects: speech recognition accuracy, meeting minutes generation quality, system response time, and user feedback.

4.1 Speech Recognition Accuracy

Speech recognition is the first step in the workflow, and its accuracy directly impacts the quality of the generated minutes. We evaluated the Whisper-large-v3 model using metrics such as Word Error Rate

(WER) and sentence recognition accuracy.

Evaluation Metrics for Whisper-large-v3:

- **Word Error Rate (WER)**: Measures the percentage of incorrect words in the transcription compared to the ground truth.
- **Sentence Accuracy**: Measures the percentage of sentences transcribed without errors.

Comparison Between Meeting Transcripts and Audio Transcriptions:

- We compared the text generated by Whisper-large-v3 with manually transcribed meeting notes under noise-free conditions.
- Results showed that Whisper-large-v3 achieved a WER of less than 5% in clear audio conditions, demonstrating excellent performance.

4.2 Meeting Minutes Generation Quality

The quality of the generated meeting minutes was evaluated through manual assessment.

Manual Evaluation Criteria:

- **Completeness**: Whether the minutes cover all key discussion points, decisions, and action items.
- **Readability**: Whether the minutes are well-structured and easy to understand.
- **Relevance**: Whether the generated content accurately reflects the meeting context.

Results:

- The generated minutes scored high in completeness and readability, accurately reflecting the meeting content.
- For complex meetings, some details were missing, but prompt engineering significantly improved the quality.

4.3 System Response Time

The response time of the system is a critical factor for user experience. We tested the system using a 12-minute sample recording on an NVIDIA RTX 3090 GPU.

Speech Recognition Phase:

- The Whisper-large-v3 model processed the 12-minute audio in approximately 1-2 minutes.

Minutes Generation Phase:

- The ChatGPT API generated the meeting minutes in about 30-60 seconds, depending on the complexity of the content.

Total Response Time:

- The entire process, from audio input to final minutes generation, took approximately 2-3 minutes, demonstrating high efficiency.

4.4 User Feedback

We invited several classmates to test the tool and collected their feedback.

Feedback Results:

- The classmates highly praised the Meeting Minutes Assistant, noting that it significantly improved the efficiency of meeting documentation.
- The generated minutes were well-structured, accurate, and met the needs of daily study and work.
- The tool's ease of use and multilingual support were also well-received.

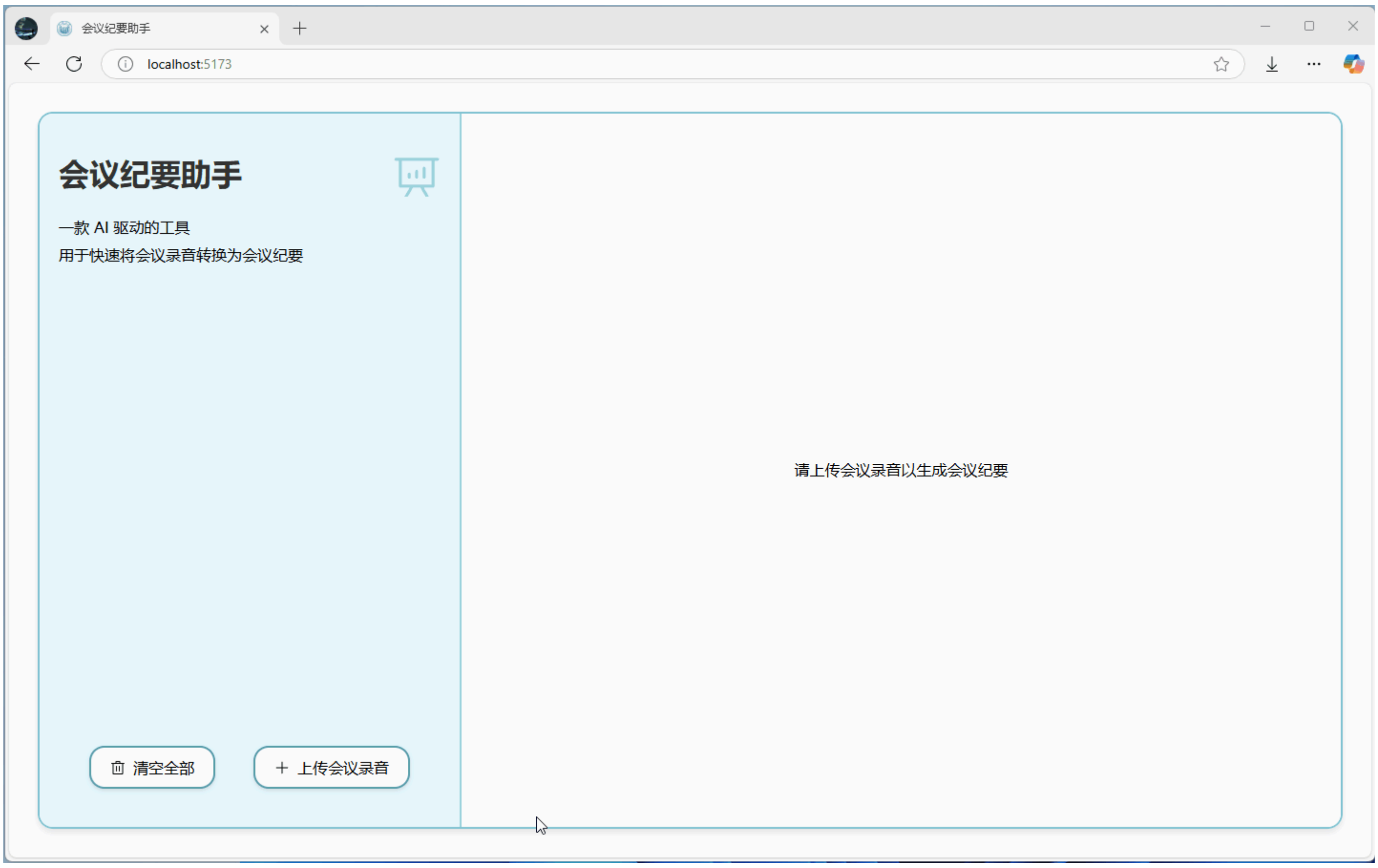
Suggestions for Improvement:

- Some classmates suggested adding more customizable templates to adapt to different meeting scenarios.
- The response time for very long meetings (e.g., over 2 hours) could be further optimized.

4.5 Conclusion

The evaluation results demonstrate that the Meeting Minutes Assistant performs exceptionally well in speech recognition accuracy, minutes generation quality, system response time, and user experience. While there is room for improvement in certain scenarios, the tool already meets the needs of most users and is a highly efficient and practical AI solution.

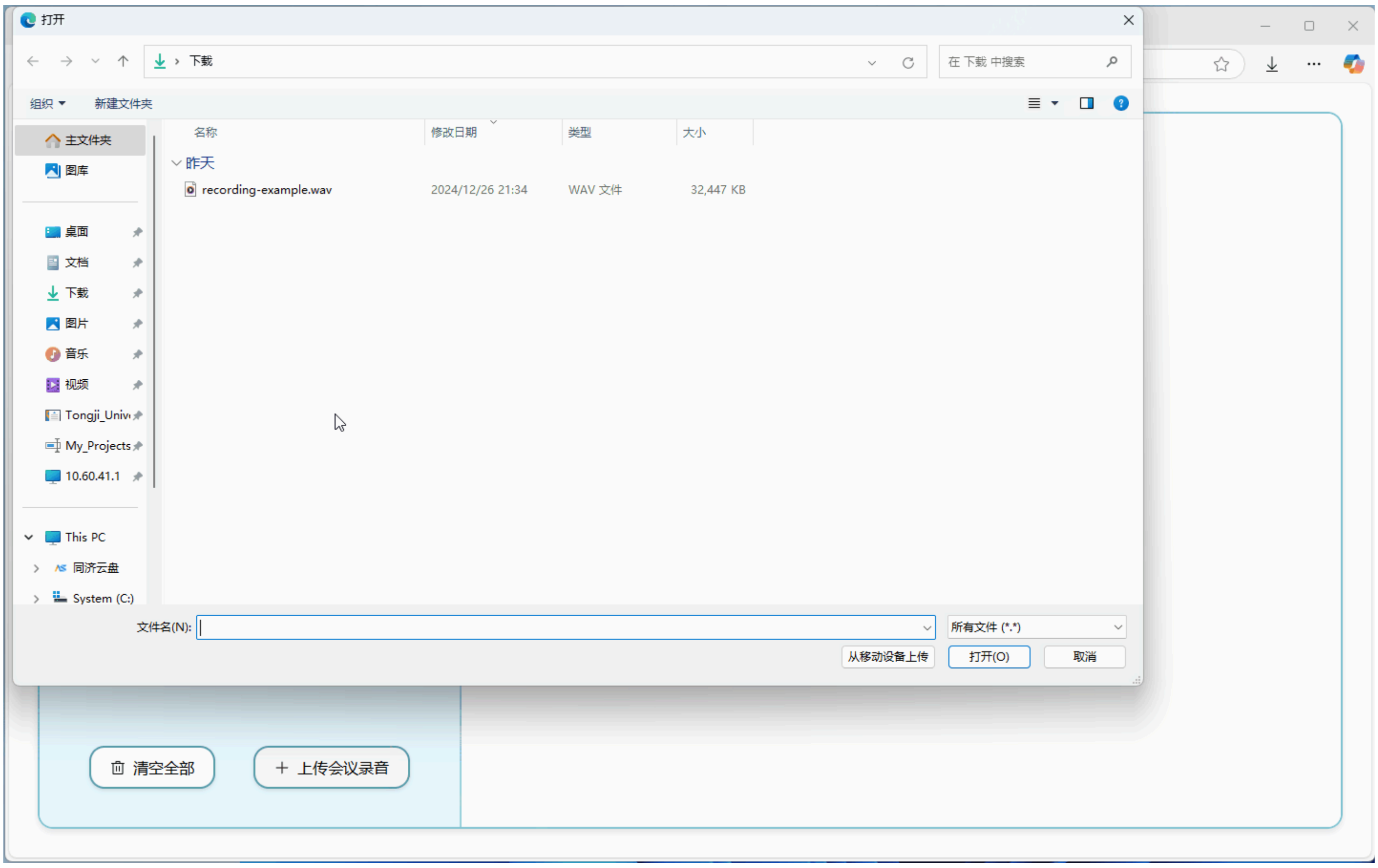
5 Project Presentation



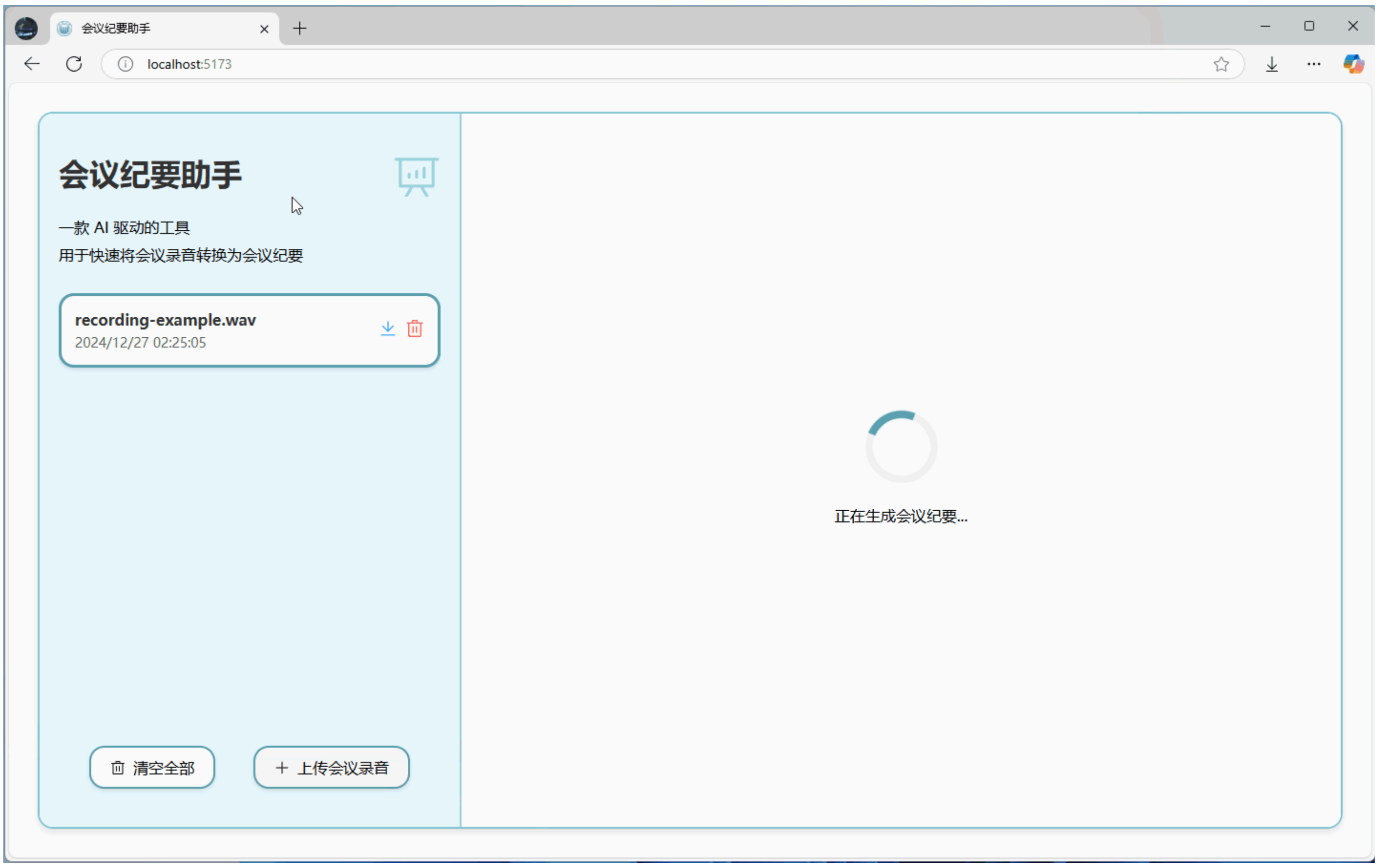
会议纪要助手

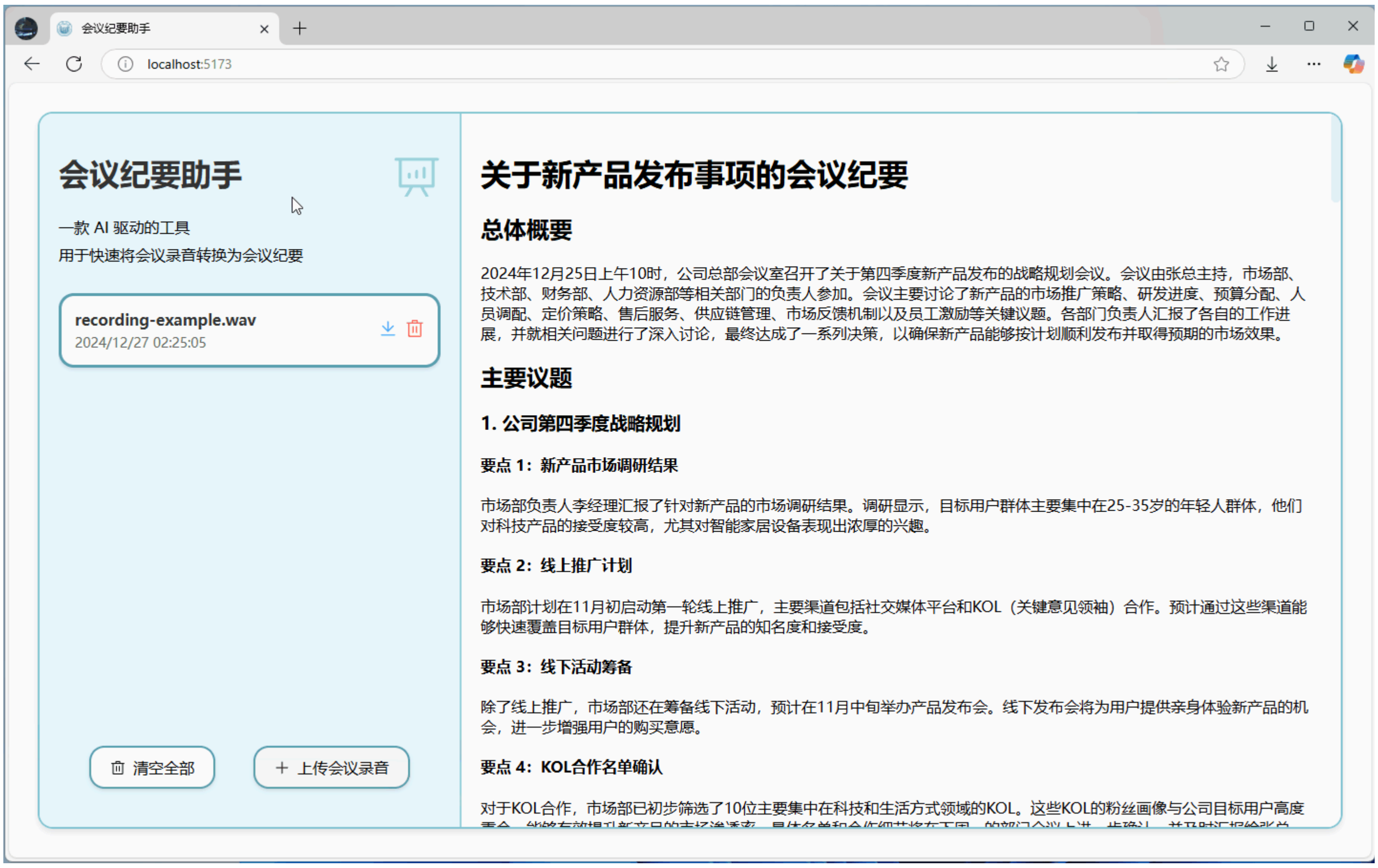
一款 AI 驱动的工具
用于快速将会议录音转换为会议纪要

请上传会议录音以生成会议纪要



名称	修改日期	类型	大小
昨天			
recording-example.wav	2024/12/26 21:34	WAV 文件	32,447 KB





关于新产品发布事项的会议纪要

总体概要

2024年12月25日上午10时，公司总部会议室召开了关于第四季度新产品发布的战略规划会议。会议由张总主持，市场部、技术部、财务部、人力资源部等相关部门的负责人参加。会议主要讨论了新产品的市场推广策略、研发进度、预算分配、人员调配、定价策略、售后服务、供应链管理、市场反馈机制以及员工激励等关键议题。各部门负责人汇报了各自的工作进展，并就相关问题进行了深入讨论，最终达成了一系列决策，以确保新产品能够按计划顺利发布并取得预期的市场效果。

主要议题

1. 公司第四季度战略规划

要点 1：新产品市场调研结果

市场部负责人李经理汇报了针对新产品的市场调研结果。调研显示，目标用户群体主要集中在25-35岁的年轻人群体，他们对科技产品的接受度较高，尤其对智能家居设备表现出浓厚的兴趣。

要点 2：线上推广计划

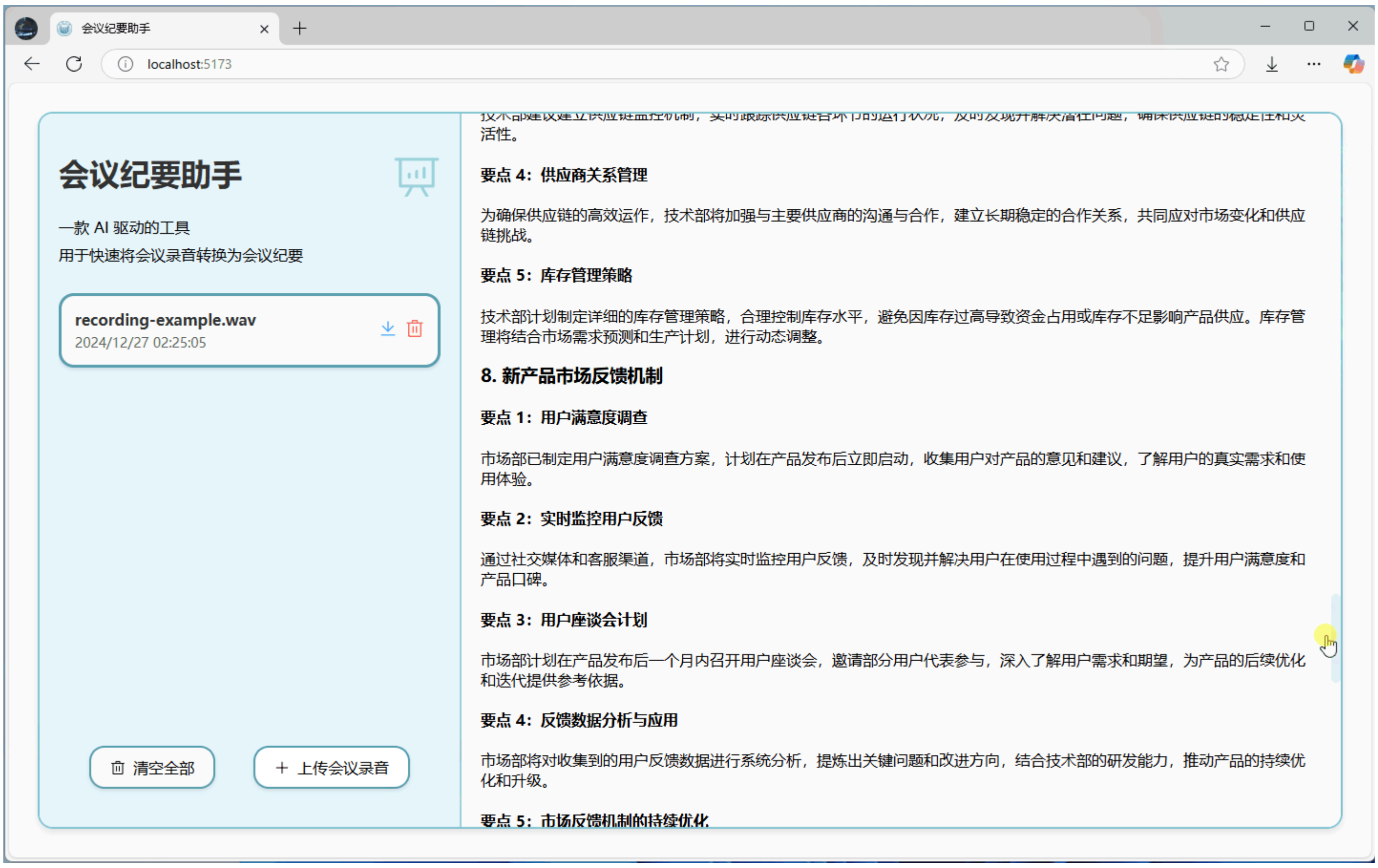
市场部计划在11月初启动第一轮线上推广，主要渠道包括社交媒体平台和KOL（关键意见领袖）合作。预计通过这些渠道能够快速覆盖目标用户群体，提升新产品的知名度和接受度。

要点 3：线下活动筹备

除了线上推广，市场部还在筹备线下活动，预计在11月中旬举办产品发布会。线下发布会将为用户提供亲身体验新产品的机会，进一步增强用户的购买意愿。

要点 4：KOL合作名单确认

对于KOL合作，市场部已初步筛选了10位主要集中在科技和生活方式领域的KOL。这些KOL的粉丝画像与公司目标用户高度重合，能够有效提升新产品的市场渗透率。具体名单和合作细节将在下周一的部门会议上进一步确认，并及时汇报给张总



会议纪要助手

一款 AI 驱动的工具
用于快速将会议录音转换为会议纪要

recording-example.wav
2024/12/27 02:25:05

活性。

要点 4：供应商关系管理

为确保供应链的高效运作，技术部将加强与主要供应商的沟通与合作，建立长期稳定的合作关系，共同应对市场变化和供应链挑战。

要点 5：库存管理策略

技术部计划制定详细的库存管理策略，合理控制库存水平，避免因库存过高导致资金占用或库存不足影响产品供应。库存管理将结合市场需求预测和生产计划，进行动态调整。

8. 新产品市场反馈机制

要点 1：用户满意度调查

市场部已制定用户满意度调查方案，计划在产品发布后立即启动，收集用户对产品的意见和建议，了解用户的真实需求和使用体验。

要点 2：实时监控用户反馈

通过社交媒体和客服渠道，市场部将实时监控用户反馈，及时发现并解决用户在使用过程中遇到的问题，提升用户满意度和产品口碑。

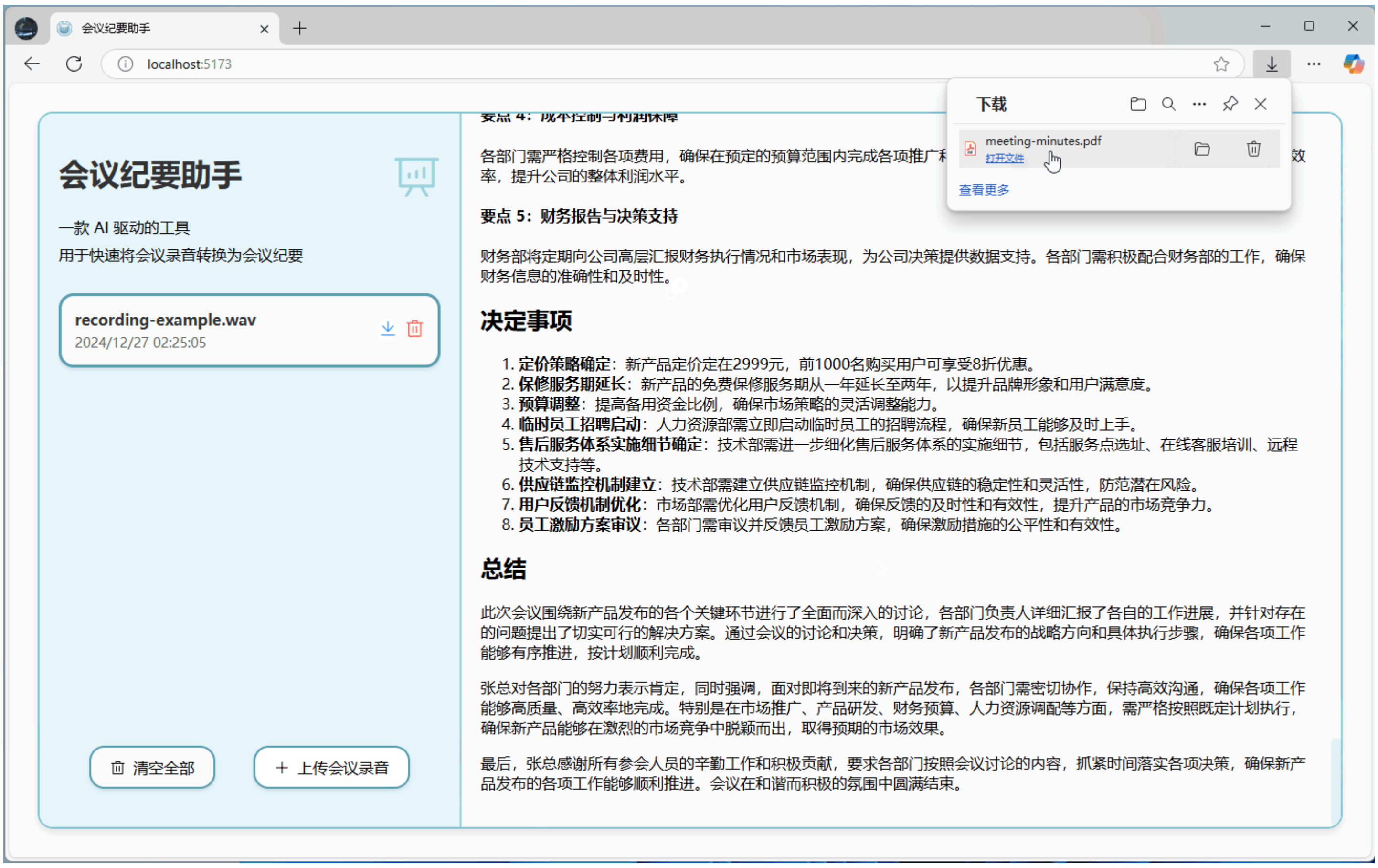
要点 3：用户座谈会计划

市场部计划在产品发布后一个月内召开用户座谈会，邀请部分用户代表参与，深入了解用户需求和期望，为产品的后续优化和迭代提供参考依据。

要点 4：反馈数据分析与应用

市场部将对收集到的用户反馈数据进行系统分析，提炼出关键问题和改进方向，结合技术部的研发能力，推动产品的持续优化和升级。

要点 5：市场反馈机制的持续优化



会议纪要助手

一款 AI 驱动的工具
用于快速将会议录音转换为会议纪要

recording-example.wav
2024/12/27 02:25:05

要点 4：成本控制与利润保障

各部门需严格控制各项费用，确保在预定的预算范围内完成各项推广和
率，提升公司的整体利润水平。

要点 5：财务报告与决策支持

财务部将定期向公司高层汇报财务执行情况和市场表现，为公司决策提供数据支持。各部门需积极配合财务部的工作，确保财务信息的准确性和及时性。

决定事项

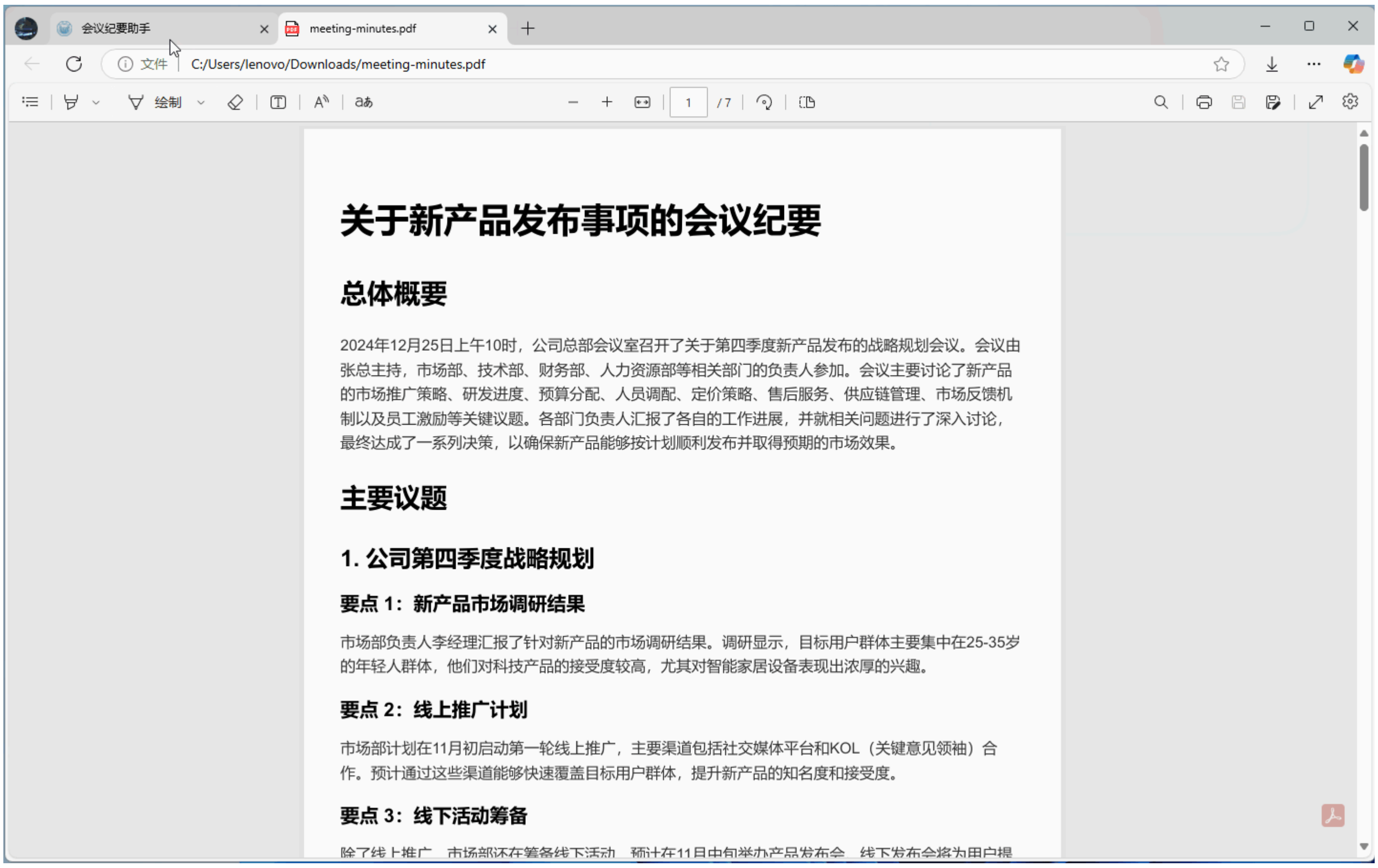
1. **定价策略确定**：新产品定价定在2999元，前1000名购买用户可享受8折优惠。
2. **保修服务期延长**：新产品的免费保修服务期从一年延长至两年，以提升品牌形象和用户满意度。
3. **预算调整**：提高备用资金比例，确保市场策略的灵活调整能力。
4. **临时员工招聘启动**：人力资源部需立即启动临时员工的招聘流程，确保新员工能够及时上手。
5. **售后服务体系实施细节确定**：技术部需进一步细化售后服务体系的实施细节，包括服务点选址、在线客服培训、远程技术支持等。
6. **供应链监控机制建立**：技术部需建立供应链监控机制，确保供应链的稳定性和灵活性，防范潜在风险。
7. **用户反馈机制优化**：市场部需优化用户反馈机制，确保反馈的及时性和有效性，提升产品的市场竞争力。
8. **员工激励方案审议**：各部门需审议并反馈员工激励方案，确保激励措施的公平性和有效性。

总结

此次会议围绕新产品发布的各个关键环节进行了全面而深入的讨论，各部门负责人详细汇报了各自的工作进展，并针对存在的问题提出了切实可行的解决方案。通过会议的讨论和决策，明确了新产品发布的战略方向和具体执行步骤，确保各项工作能够有序推进，按计划顺利完成。

张总对各部门的努力表示肯定，同时强调，面对即将到来的新产品发布，各部门需密切协作，保持高效沟通，确保各项工作能够高质量、高效率地完成。特别是在市场推广、产品研发、财务预算、人力资源调配等方面，需严格按照既定计划执行，确保新产品能够在激烈的市场竞争中脱颖而出，取得预期的市场效果。

最后，张总感谢所有参会人员的辛勤工作和积极贡献，要求各部门按照会议讨论的内容，抓紧时间落实各项决策，确保新产品发布的各项工作能够顺利推进。会议在和谐而积极的氛围中圆满结束。



关于新产品发布事项的会议纪要

总体概要

2024年12月25日上午10时，公司总部会议室召开了关于第四季度新产品发布的战略规划会议。会议由张总主持，市场部、技术部、财务部、人力资源部等相关部门的负责人参加。会议主要讨论了新产品的市场推广策略、研发进度、预算分配、人员调配、定价策略、售后服务、供应链管理、市场反馈机制以及员工激励等关键议题。各部门负责人汇报了各自的工作进展，并就相关问题进行了深入讨论，最终达成了一系列决策，以确保新产品能够按计划顺利发布并取得预期的市场效果。

主要议题

1. 公司第四季度战略规划

要点 1：新产品市场调研结果

市场部负责人李经理汇报了针对新产品的市场调研结果。调研显示，目标用户群体主要集中在25-35岁的年轻人群体，他们对科技产品的接受度较高，尤其对智能家居设备表现出浓厚的兴趣。

要点 2：线上推广计划

市场部计划在11月初启动第一轮线上推广，主要渠道包括社交媒体平台和KOL（关键意见领袖）合作。预计通过这些渠道能够快速覆盖目标用户群体，提升新产品的知名度和接受度。

要点 3：线下活动筹备

除了线上推广，市场部还在筹备线下活动，预计在11月中旬举办产品发布会，线下发布会将为用户提

6 Advantages and Disadvantages of the Method

6.1 Advantages

1. **High Efficiency** The Meeting Minutes Assistant significantly reduces the time required to generate meeting minutes, transforming a task that typically takes hours into one that takes just a few minutes.
2. **Accuracy and Completeness**: Leveraging state-of-the-art models like Whisper-large-v3 and ChatGPT ensures high accuracy in both speech recognition and text summarization, minimizing information loss.
3. **Structured Output**: The tool generates well-organized meeting minutes, including key elements such as discussion points, decisions, and action items, making it easy for users to review and follow up.
4. **Multilingual Support**: The ability to handle multiple languages makes the tool suitable for international teams and diverse work environments.
5. **User-Friendly Interface**: The modular design and intuitive workflow lower the barrier to entry, allowing users with minimal technical expertise to operate the tool effectively.
6. **Customizability**: Through prompt engineering and customizable templates, users can tailor the output to meet specific needs and preferences.
7. **Local Deployment**: The use of locally deployed Whisper-large-v3 ensures data privacy and reduces dependency on cloud services, which is particularly important for sensitive information.

6.2 Disadvantages

1. **Dependence on Audio Quality**: The accuracy of speech recognition is highly dependent on the quality of the audio recording. Noisy environments or poor recording equipment can degrade performance.
2. **Cost of API Usage**: While Whisper-large-v3 is open-source and locally deployable, the use of OpenAI’s ChatGPT API incurs costs, which may be a concern for large-scale or frequent usage.
3. **Handling Complex Meetings**: For highly complex meetings with multiple speakers or overlapping discussions, the tool may struggle to capture all details accurately, requiring manual intervention.
4. **Response Time for Long Meetings**: While the tool is efficient for short to medium-length meetings, the response time for very long meetings (e.g., over 2 hours) can still be improved.

7 Conclusion

The Meeting Minutes Assistant is an innovative AI-powered tool designed to automate the generation of meeting minutes, significantly improving efficiency and accuracy. By integrating advanced speech recognition (Whisper-large-v3) and text generation (ChatGPT) models, the tool quickly converts meeting recordings into structured and actionable minutes. It excels in handling multiple languages, producing well-organized outputs, and offering a user-friendly experience, making it suitable for diverse teams and work environments.

Despite its strengths, the tool has some limitations, such as its dependence on audio quality, the cost of API usage, and challenges in handling highly complex or lengthy meetings. However, these issues present opportunities for future enhancements, such as improving noise robustness, exploring open-source alternatives, and optimizing performance for longer recordings.

Overall, the Meeting Minutes Assistant is a practical and powerful solution that addresses a critical need in modern workplaces. By reducing the time and effort required for meeting documentation, it empowers users to focus on more valuable tasks, fostering productivity and collaboration. With continued development, the tool has the potential to become an essential asset for teams and organizations worldwide.