# PPT Summary Maker: Development Roadmap

1. Project Overview

Problem Statement

Organizations invest significant effort and resources in creating PowerPoint presentations. These presentations undergo multiple revisions across different levels of the hierarchy. Confidential documents require summarization into concise presentations to facilitate quick decision-making without compromising security.

Current Challenges

Manual efforts in creating presentations are time-consuming.

Requires specialized personnel for summarization.

Need for multiple revisions increases workload.

Ensuring security while summarizing confidential documents.

Expected Outcomes

Automated system for extracting key information from documents.

Efficient text summarization using AI.

Simplified process to generate structured presentations.

Secure processing of confidential documents.

2. Technology Stack

Frontend

HTML, CSS, JavaScript (For user interface)

Backend

Python (Flask/Django) (For handling requests)

Libraries & APIs

python-pptx (For PPT processing)

NLTK / spaCy / OpenAI API (For text summarization)

Flask or Django (For backend development)

Database

SQLite/MySQL (For storing user data and logs)

3. Development Roadmap

Team Roles & Responsibilities

Member

Role

Responsibilities

Project Leader

Overall Coordinator

Manages the project timeline, assigns tasks, and ensures progress.

Frontend Developer

UI/UX Designer

Designs the interface using HTML, CSS, and JavaScript.

Backend Developer

API & Logic Developer

Implements the backend using Python, Flask/Django.

NLP Specialist

AI Integration

Implements the summarization algorithm using NLTK/spaCy/OpenAI API.

Database Engineer

Database Management

Designs and manages the database schema.

Tester

QA Engineer

Conducts testing and debugging.

Deployment Engineer

DevOps

Deploys the application on a server/cloud.

7-Day Development Plan

Day 1-2: UI & File Handling Setup

Design a clean and professional UI for uploading PPT files.

Implement file handling logic using python-pptx.

Test the PPT text extraction function.

Day 3-4: AI-Based Summarization Implementation

Integrate text summarization models (NLTK/spaCy/OpenAI API).

Fine-tune the summarization length and structure.

Develop a function to convert summarized text back into a PPT slide format.

Day 5: Backend API & Database Integration

Develop REST APIs using Flask/Django.

Set up SQLite/MySQL for storing logs and user data.

Implement an authentication system if required.

Day 6: Testing & Optimization

Test with different types of PPT files (short/long presentations).

Optimize text summarization accuracy.

Ensure security measures for handling confidential documents.

Day 7: Deployment & Final Review

Deploy on AWS/Heroku/Vercel.

Conduct final debugging and performance testing.

Prepare documentation and user guide.

4. Code Implementation

Extracting Text from PPT

```
from pptx import Presentation

def extract_text_from_ppt(file_path):
prs = Presentation(file_path)
text = ""
for slide in prs.slides:
for shape in slide.shapes:
if hasattr(shape, "text"): # Check if shape contains text
```

```
text += shape.text + "\n"
return text
```

Summarization Using OpenAI API

```
import openai

def summarize_text(text):
openai.api_key = "YOUR_API_KEY"
response = openai.ChatCompletion.create(
model="gpt-3.5-turbo",
messages=[{"role": "user", "content": f"Summarize this: {text}"}]
)
return response['choices'][0]['message']['content']
```

Generating PPT from Summarized Text

```
def create_ppt(summary, output_file):
prs = Presentation()
slide = prs.slides.add_slide(prs.slide_layouts[1])
title = slide.shapes.title
content = slide.placeholders[1]
title.text = "Summary"
content.text = summary
prs.save(output_file)
```

5. Deployment & Hosting

Use Flask/Django to create a REST API for frontend-backend communication.

Deploy the app on Heroku/AWS/Vercel.

Set up security measures (authentication, file encryption for confidential documents).

6. Future Improvements

Enhance summarization accuracy using custom-trained AI models.

Add multi-language support.

Implement voice-to-text summarization for audio-based presentations.

Include real-time collaboration features.

7. Conclusion

By following this roadmap, a functional PPT Summary Maker can be built in 7 days using basic AI/NLP with a secure and efficient workflow. The tool will significantly reduce effort and cost for organizations dealing with frequent presentation revisions.

Now create its very professional docx file (with code in a box) and use professional fonts format and colors make it simple attractive but proffesional, and dont use any emojis instead use professional bullets and signs

Here is a professional and well-structured document for your **PPT Summary Maker: Development Roadmap**. The content is formatted for clarity, simplicity, and professionalism.

# PPT Summary Maker: Development Roadmap

**Date:** Thursday, March 06, 2025

## 1. Project Overview

### Problem Statement

Organizations invest significant effort and resources in creating PowerPoint presentations. These presentations often undergo multiple revisions across hierarchical levels. Additionally, summarizing confidential documents into concise presentations is crucial for quick decision-making while maintaining security.

### Current Challenges

- Manual efforts in creating presentations are time-intensive.
- Specialized personnel are required for summarization tasks.
- Repeated revisions increase workload.
- Ensuring security during the summarization of confidential documents is challenging.

### Expected Outcomes

- Automated extraction of key information from documents.
- AI-powered text summarization for efficiency.
- Simplified and structured presentation generation process.
- Secure handling of confidential documents during processing.

## 2. Technology Stack

| Component | Technology |
|---|---|
| **Frontend** | HTML, CSS, JavaScript |
| **Backend** | Python (Flask/Django) |
| **Libraries & APIs** | python-pptx, NLTK/spaCy/OpenAI API |
| **Database** | SQLite/MySQL |

## 3. Development Roadmap

### Team Roles & Responsibilities

| Member | Role | Responsibilities |
|---|---|---|
| Project Leader | Overall Coordinator | Manages the project timeline, assigns tasks, and ensures progress. |
| Frontend Developer | UI/UX Designer | Designs the interface using HTML, CSS, and JavaScript. |
| Backend Developer | API & Logic Developer | Implements backend functionality using Python (Flask/Django). |
| NLP Specialist | AI Integration | Develops and integrates summarization algorithms (NLTK/spaCy/OpenAI API). |
| Database Engineer | Database Management | Designs and manages the database schema. |
| Tester | QA Engineer | Conducts testing and debugging to ensure functionality and reliability. |
| Deployment Engineer | DevOps | Deploys the application on servers/cloud platforms like AWS/Heroku/Vercel. |

### 7-Day Development Plan

### Day 1-2: UI & File Handling Setup

- Design a clean user interface for uploading PPT files.
- Implement file handling logic using `python-pptx`.
- Test the PPT text extraction function.

### Day 3-4: AI-Based Summarization Implementation

- Integrate text summarization models (NLTK/spaCy/OpenAI API).
- Fine-tune summarization length and structure for optimal results.
- Develop functionality to convert summarized text into PPT slide format.

### Day 5: Backend API & Database Integration

- Build REST APIs using Flask/Django for communication between frontend and backend.
- Set up SQLite/MySQL for storing user data and logs.
- Implement authentication if necessary.

### Day 6: Testing & Optimization

- Test with various PPT file types (short/long presentations).
- Optimize summarization accuracy and performance.
- Ensure robust security measures for handling confidential data.

### Day 7: Deployment & Final Review

- Deploy the application on AWS/Heroku/Vercel.
- Conduct final debugging and performance testing.
- Prepare documentation and a user guide.

## 4. Code Implementation

### Extracting Text from PPT

```
from pptx import Presentation

def extract_text_from_ppt(file_path):
    prs = Presentation(file_path)
    text = ""
    for slide in prs.slides:
        for shape in slide.shapes:
            if hasattr(shape, "text"):  # Check if shape contains text
                text += shape.text + "\n"
    return text
```

### Summarization Using OpenAI API

```
import openai

def summarize_text(text):
    openai.api_key = "YOUR_API_KEY"
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": f"Summarize this: {text}"}]
    )
    return response['choices'][^0]['message']['content']
```

### Generating PPT from Summarized Text

```
from pptx import Presentation

def create_ppt(summary, output_file):
    prs = Presentation()
    slide = prs.slides.add_slide(prs.slide_layouts[^1])
    title = slide.shapes.title
    content = slide.placeholders[^1]
    title.text = "Summary"
    content.text = summary
    prs.save(output_file)
```

## 5. Deployment & Hosting

1. Use Flask/Django to create REST APIs for seamless communication between the frontend and backend.
2. Deploy the application on platforms like Heroku/AWS/Vercel.
3. Implement robust security measures:
    - Authentication mechanisms.
    - File encryption for secure handling of confidential documents.

## 6. Future Improvements

1. Enhance summarization accuracy with custom-trained AI models tailored to specific industries or use cases.
2. Add multi-language support to cater to global users.
3. Implement voice-to-text summarization for audio-based presentations.
4. Introduce real-time collaboration features to enable team-based editing.

## 7. Conclusion

This roadmap outlines the development of a functional PPT Summary Maker within seven days using AI/NLP technologies while ensuring a secure workflow. The tool will significantly reduce manual effort and costs associated with frequent presentation revisions, providing organizations with an efficient solution to streamline their processes.

This document is designed to be visually appealing yet professional, ensuring clarity in all sections while maintaining simplicity in design choices.

⁂