Project Report: AI-Powered Meeting Notes Summarizer and Sharer

Introduction

This report details the development, approach, process, and tech stack for building an AI-powered meeting notes summarizer and sharer application. The project was designed to enable users to upload text transcripts, input custom prompts (e.g., "Summarize in bullet points"), generate AI-structured summaries, edit them, and share via email with multiple recipients. The solution was tailored for a Windows environment using Command Prompt and File Explorer, with a focus on functionality over aesthetics, and deployed on Render.com.

Approach

The development followed an iterative, problem-solving approach, adapting to real-time challenges (e.g., API quota limits, deployment issues). The process began with a local setup, transitioned to integrating an AI service, resolved dependency conflicts, and culminated in deployment. Key decisions included:

- **Modularity**: Separating frontend and backend for independent testing and deployment.
- **API Flexibility**: Switching from OpenAI to Hugging Face due to quota limits, ensuring scalability with a free tier.
- **User-Centric Testing**: Using 10 diverse .txt files to validate all use cases, including edge cases like empty transcripts.
- **Deployment Focus**: Prioritizing Render's free tier for accessibility, with GitHub as the version control backbone.

The approach was guided by troubleshooting logs and user feedback, ensuring a functional end product.

Process

1. **Project Setup**:
    - Created a directory structure (meeting-summarizer/backend, meeting-summarizer/frontend) using Command Prompt.
    - Initialized Git for version control, excluding node_modules and .env via .gitignore.
    - Configured environment variables (HUGGINGFACE_API_KEY, SENDGRID_API_KEY, PORT) in a .env file.

2. **Backend Development**:
    - Built with Node.js and Express, using Multer for file uploads and Nodemailer with SendGrid for email.
    - Integrated Hugging Face Inference API (facebook/bart-large-cnn model) for summarization after encountering OpenAI quota limits.
    - Added error handling for empty transcripts, returning "No content to summarize".
    - Tested endpoints (/generate-summary, /send-email) with Postman.
3. **Frontend Development**:
    - Created a minimal UI with HTML, vanilla JavaScript, and CSS, focusing on form submission, summary display, and email sharing.
    - Implemented fetch requests to communicate with the backend, updating URLs for deployment.
4. **Testing**:
    - Developed 10 .txt files (e.g., long_multi_section.txt, empty_transcript.txt) with varied prompts to cover summarization styles (bullet points, action items, executive overviews) and edge cases.
    - Verified functionality locally and post-deployment, ensuring multiple email support.
5. **Deployment**:
    - Pushed code to GitHub using Git.
    - Deployed backend as a Node.js web service and frontend as a static site on Render.com.
    - Configured environment variables in Render and updated frontend fetch URLs to the backend URL (e.g., https://meeting-summarizer-8kjq.onrender.com).

Tech Stack of Choice

- **Frontend**:
    - **HTML5**: For basic structure and form elements.
    - **Vanilla JavaScript**: For client-side logic (fetch requests, DOM manipulation).
    - **CSS**: Minimal styling for usability.

    - **Rationale**: Kept simple to meet project requirements, avoiding frameworks for speed and simplicity.

- **Backend**:
    - **Node.js**: Runtime for server-side logic, version 22.x on Render.
    - **Express.js (v4.19.2)**: Web framework for routing and middleware.
    - **Multer (v1.4.5-lts.1)**: File upload handling for transcripts.
    - **Nodemailer (v6.9.13) with SendGrid**: Email service for sharing summaries.
    - **@huggingface/inference (v2.5.2)**: AI summarization via the facebook/bart-large-cnn model.
    - **dotenv (v17.2.1)**: Environment variable management.
    - **cors (v2.8.5)**: Cross-origin resource sharing for frontend-backend communication.
    - **Rationale**: Chosen for compatibility, free-tier availability, and proven stability in similar projects.
- **Version Control and Deployment**:
    - **Git**: For version control, hosted on GitHub.
    - **Render.com**: Hosting platform for free-tier Node.js and static site deployment.
    - **Rationale**: GitHub provides a free, accessible repository, while Render offers easy deployment with environment variable support.

Conclusion

The project successfully delivered a functional AI-powered summarizer and sharer, overcoming initial API constraints and deployment hurdles. The chosen tech stack balanced performance, cost, and ease of use, while the process ensured thorough testing and scalability. The deployed application, accessible via its frontend URL, meets the core requirements and is ready for further enhancement or production use.