# Project Documentation: AI Meeting Summarizer

Author: Sujithra Dhayalan

Date: August 17, 2025

Live Application: [Live demo](https://sum-it-up-ai-meeting-summarizer-one.vercel.app/)

GitHub Repository: [Github](https://github.com/Sujithra-dhayalan/Sum-It-Up)

### 1. Project Overview

This project is a full-stack AI-powered application designed to summarize meeting transcripts. It allows a user to input a text transcript and a custom prompt, generate a structured summary using an AI model, edit the result, and share it via email.

The core focus of this project was to build a functional, end-to-end application emphasizing robust backend functionality over a complex user interface.

### 2. My Approach and Process

Given the 24-hour deadline, my process was centered on creating a Minimum Viable Product (MVP) that met all core requirements, prioritizing functionality and speed.

1. **Requirement Analysis:** I first broke down the project requirements into key features: text input, prompt input, AI generation, editable output, and email sharing.
2. **Architectural Decision:** I opted for a decoupled architecture with a React frontend and a Node.js/Express backend. This separation of concerns allows for independent development, scaling, and deployment. Crucially, I decided to **omit a database** for this initial version, as data persistence was not a core requirement. This strategic simplification was key to completing the project on time.
3. **Development Workflow:**
   * **Backend First:** I began by building the Express API with two essential endpoints: /api/summarize to handle the AI logic with the *Groq API*, and /api/share to manage email sending with *Nodemailer*.
   * **Frontend Next:** With the backend logic in place, I built a minimal React UI to interact with these endpoints, focusing on a clean and functional user flow.
   * **Deployment:** I chose Render for the backend and Vercel for the frontend due to their excellent Git integration, generous free tiers, and straightforward setup process.

### 3. Tech Stack & Justification

| **Component** | **Technology** | **Justification** |
| --- | --- | --- |
| **Frontend** | React | Chosen for its component-based architecture and vast ecosystem, allowing for rapid UI development. |
| **Backend** | Node.js & Express | Ideal for creating a lightweight, fast, and scalable API. Using JavaScript across the stack improves development speed. |
| **AI Service** | Groq API | Selected specifically for its industry-leading inference speed, providing an excellent and responsive user experience. |
| **Email** | Nodemailer & Gmail API | Nodemailer is the standard for sending emails in Node.js. The Gmail API with an App Password was used for its reliability. |
| **Deployment** | Vercel & Render | This combination is a best-in-class solution for deploying MERN-stack applications, offering seamless CI/CD and hassle-free configuration. |

### 4. Challenges & Solutions

**Challenge 1: Email Deliverability**

* **Problem:** During development, emails sent via the application were not being received, despite the server logs indicating success. Initial attempts with a third-party SMTP service (Brevo) resulted in authentication errors and deliverability issues.
* **Solution:** I systematically debugged the issue by enhancing the server logs to inspect the SMTP response. After isolating the authentication problem, I pivoted to a more reliable solution: using Google's SMTP server via Nodemailer. This required enabling 2-Step Verification on a Google account and generating a secure 16-digit **App Password**, which resolved all authentication and deliverability problems.

**Challenge 2: Meeting the Tight Deadline**

* **Problem:** Building a full-stack application with AI and email integration in under 24 hours required careful planning and prioritization.
* **Solution:** I focused strictly on the MVP requirements. By intentionally scoping out features like databases and user authentication, I was able to dedicate my time to building and debugging the core, required functionality. I chose familiar and efficient tools (like create-react-app, Express, and Vercel/Render) to minimize setup time and focus on the business logic.

### 5. Conclusion

This project successfully demonstrates the ability to rapidly design, build, and deploy a functional full-stack application. It showcases skills in API integration (Groq, Nodemailer), backend and frontend development, and modern deployment practices. The challenges faced, particularly with email authentication, highlight a capacity for methodical debugging and problem-solving under pressure.