A PROJECT REPORT

ON

VOICE-CONTROLLED INTELLIGENT TASK MANAGER

Submitted in partial fulfillment of the requirements for the internship at

GWING

SUBMITTED BY: Bryson Lopes

DATE: December 23, 2025

ACKNOWLEDGEMENT

I would like to express my gratitude to the management of **Gwing** for providing me with this opportunity to enhance my technical skills. I am deeply grateful to my mentors for their guidance, encouragement, and valuable feedback throughout the development of this project.

Their support in helping me understand the complexities of the Web Speech API and GraphQL architecture was instrumental in the successful completion of this voice-driven application.

TABLE OF CONTENTS

1. ABSTRACT

In the fast-paced digital environment fostered at **Gwing**, efficiency and user experience are paramount. This project, **"Voice-Controlled Intelligent Task Manager,"** was developed to explore the intersection of Natural Language Processing (NLP) and modern web architectures.

The application allows users to manage tasks entirely through voice commands ("Add task...", "Delete task..."), eliminating the friction of manual typing. Unlike traditional REST API applications, this project utilizes a **GraphQL** architecture to prevent data over-fetching and ensure efficient state management. Furthermore, it integrates **Google's Gemini Generative AI** to intelligently break down complex user goals into actionable sub-tasks automatically.

This report documents the end-to-end development process, from the initial architectural decisions to the final deployment using a MERN (MongoDB, Express, React, Node) stack enhanced with Apollo Federation.

CHAPTER 2: TECHNOLOGY STACK

The project utilizes a cutting-edge "Gwing-approved" stack designed for scalability and performance.

2.1 Frontend & UI

- **React.js (v18):** For building a component-based Single Page Application (SPA).
- **Tailwind CSS:** For rapid, utility-first styling.
- **Glassmorphism UI:** A modern design aesthetic using background blurs and semi-transparent gradients to provide a premium look.

2.2 Backend & Middleware

- **Node.js & Express:** Provides the runtime environment for the server.
- **GraphQL:** Replaces traditional REST endpoints, allowing the client to request specific data fields.
- **Apollo Client/Server:** Manages the data graph, caching, and loading states seamlessly.

2.3 Database & AI

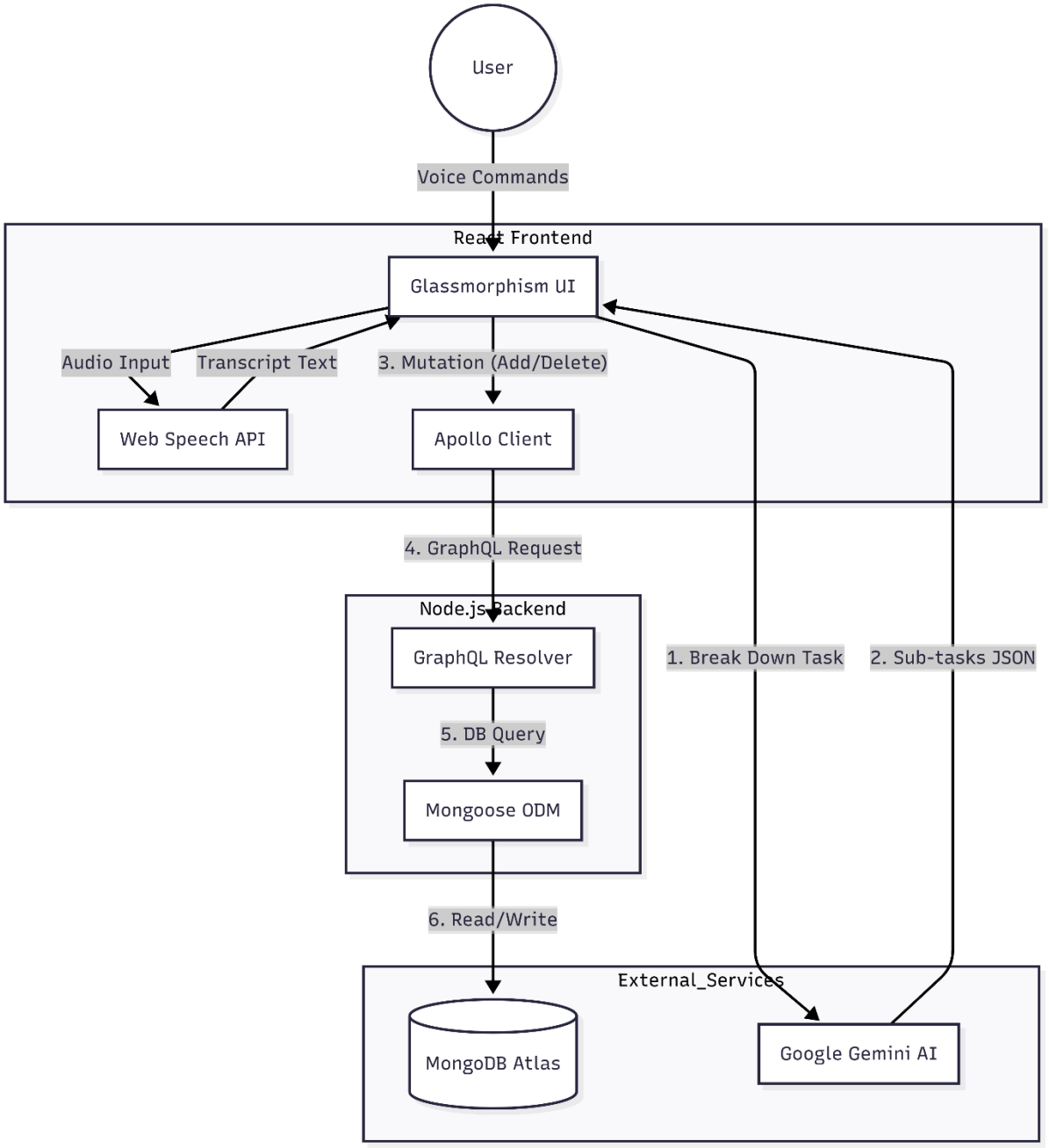
- **MongoDB Atlas:** A cloud-native NoSQL database used to store tasks as JSON documents.
- **Google Gemini API (Flash 2.5):** An LLM used to analyze task text and generate sub-task arrays JSON format.
- **Web Speech API:** A native browser interface used for Speech-to-Text (Recognition) and Text-to-Speech (Synthesis).

CHAPTER 3: SYSTEM ARCHITECTURE & DESIGN

3.1 High-Level Architecture

The system follows a decoupled client-server architecture. The Client (React) communicates with the Server (Node) exclusively via GraphQL queries and mutations. The Client also interacts directly with the Browser's Audio Hardware and the external AI Service.

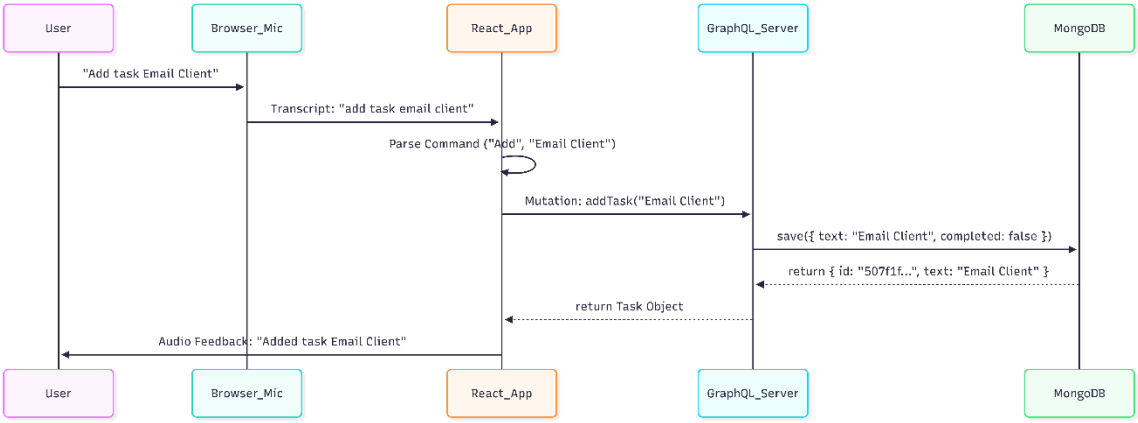
Architecture Diagram:



3.2 Data Flow Diagram (DFD)

This diagram illustrates how a voice command travels through the system to become a stored database entry.

sequenceDiagram



3.3 Database Schema

We utilized a flexible NoSQL schema to allow for future scalability at Gwing.

Collection: tasks

- _id: ObjectId (Primary Key)
- text: String (The task description)
- completed: Boolean (Status flag)
- createdAt: Timestamp

CHAPTER 4: IMPLEMENTATION

4.1 Voice Recognition Engine

The core innovation is the integration of window.SpeechRecognition. The app listens for specific keywords to trigger actions.

- **Keyword "Add task [text]":** Captures the substring after "task" and triggers the ADD_TASK mutation.
- **Keyword "Delete task [number]":** Parses the number, maps it to the array index, finds the MongoDB _id, and triggers DELETE_TASK.

4.2 GraphQL Integration

Unlike REST, where we might hit /api/tasks and get unnecessary data, our GraphQL query is precise:

```
# The exact query used in the application

query GetTasks {

  tasks {
```

```
  id
  text
  completed
 }
}
```

5.1 Task Schema

Each task contains:

- Title
- Description
- Completion status
- Creation timestamp
- Update timestamp

MongoDB is chosen because it is flexible, scalable, and easy to integrate with Node.js.

5.2 ERROR HANDLING

The application handles errors such as:

- Microphone permission denial
- Unsupported browser
- Speech recognition failure
- Network errors
- GraphQL errors

Users receive clear messages when something goes wrong.

6. ACCESSIBILITY FEATURES

- Voice-based input and output
- Keyboard-accessible buttons
- ARIA labels for screen readers
- Clear visual feedback
- Hands-free operation support

These features make the application usable for a wider audience.

6.1 SECURITY CONSIDERATIONS

- Input validation
- Controlled API access
- Safe GraphQL mutations
- No direct database exposure

Authentication can be added as a future enhancement.

6.2 Manual Testing

- Voice command testing
- UI interaction testing
- GraphQL mutation testing
- Error scenario testing

6.3 Browser Testing

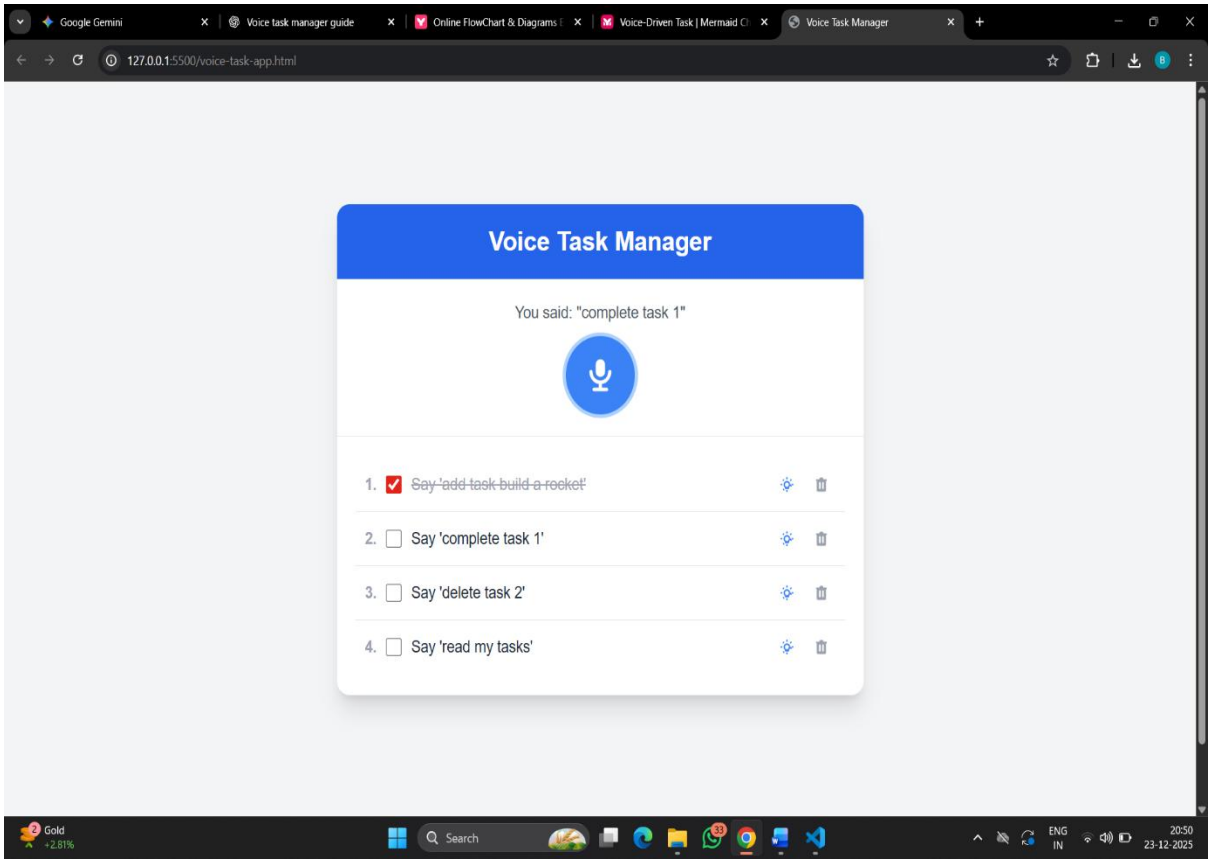
- Chrome (fully supported)
- Limited support on other browsers

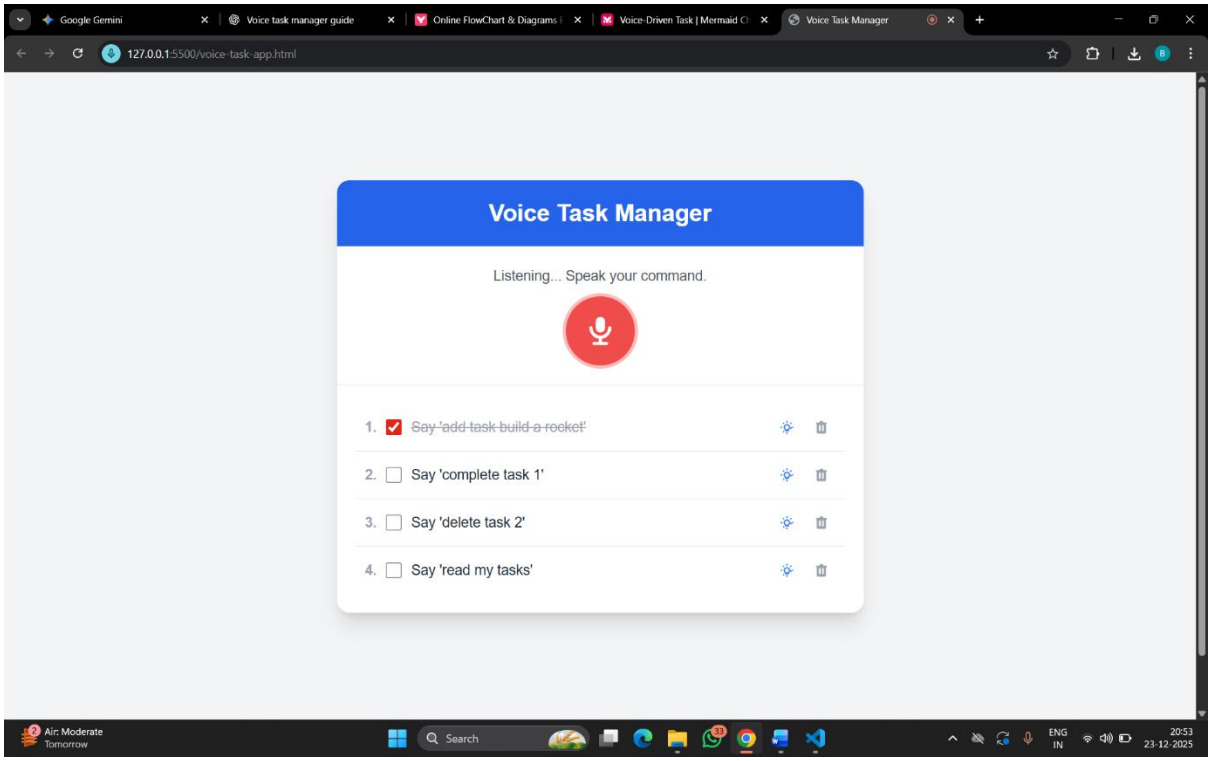
7.1 RESULTS

The application successfully:

- Accepts voice commands
- Creates and manages tasks
- Reads task summaries aloud
- Stores data in MongoDB
- Provides responsive UI

The project meets all stated objectives.


Google Gemini
Voice task manager guide
Online FlowChart & Diagrams
Voice-Driven Task | Mermaid
Voice Task Manager
127.0.0.1:5500/voice-task-app.html
Voice Task Manager
You said: "complete task 1"
1. Say 'add task build a rocket'
2. Say 'complete task 1'
3. Say 'delete task 2'
4. Say 'read my tasks'
Gold
+2.81%
Search
ENG
IN
20:50
23-12-2025


Google Gemini
Voice task manager guide
Online FlowChart & Diagrams
Voice-Driven Task | Mermaid
Voice Task Manager
127.0.0.1:5500/voice-task-app.html
Voice Task Manager
Listening... Speak your command.
1. Say 'add task build a rocket'
2. Say 'complete task 1'
3. Say 'delete task 2'
4. Say 'read my tasks'
Air: Moderate
Tomorrow
Search
ENG
IN
20:53
23-12-2025

7.2 LIMITATIONS

- Speech recognition depends on browser support
- Limited natural language understanding
- Requires internet connection
- No user authentication

7.3 FUTURE ENHANCEMENTS

- User authentication
- Advanced NLP for voice commands
- Multi-language support
- Offline mode
- Mobile application version
- Task prioritization
- Reminder notifications

8.CONCLUSION:

This internship project at **Gwing** has demonstrated the viability of voice-first web applications. By moving away from traditional input methods and legacy API structures, we have created a tool that feels futuristic and highly efficient.

Key Achievements:

1. **Zero-Touch Interface:** Users can manage their daily workload without touching the keyboard.
2. **Smart Assistance:** The AI integration transforms the app from a simple list into an intelligent assistant.
3. **Modern Engineering:** The use of GraphQL and React Hooks ensures the codebase is maintainable and scalable.

This project lays the foundation for future "Smart Office" tools at Gwing, potentially integrating with calendar APIs and team collaboration features.