**COMP3207 Cloud Application Development**

**Task 4: Application Security**

**Student ID: 31486959**

# Application Information

## Application Link

<https://main.d1ca5coxtgdx3v.amplifyapp.com>

## Source Code

Backend: <https://github.com/ThhmSamuel/meet-summarizer-backend.git>

Frontend: <https://github.com/ThhmSamuel/meet-summarizer-frontend.git>

## Demo Account Information

Email: [test@gmail.com](mailto:test@gmail.com)

Password: 123456

# Meet Summarizer AI Application Security Documentation

## Introduction

The Meeting Minutes AI application places security at the core of its design by having user data protection, authenticated access, and confidentiality maintained on the system. This document describes the application’s security architecture, explaining how user data is safeguarded in different stages such as during storage and transmission, explaining the authentication processes, as well as detailing how the system is protected within the cloud infrastructure.

## Security Measures for Data

Some of the sensitive data processed by the application includes audio recordings, transcription files, and associated user credentials. Each data type is protected according to its sensitivity and relevance, obtaining distinctive measures tailored to its requirements.

## Handling of Audio Files

When audio recordings of meetings are received from a user, the audio file will be handled along a pre-defined workflow designed to keep it secure. The audio file is first handled by the frontend React app and then uploaded to the backend server over a secure HTTPS channel. This upload is encapsulated in the uploadAndProcessAudio method of the frontend API service, which constructs the file as a FormData object:

A screen shot of a computer program

AI-generated content may be incorrect.

On the server’s side, the file is handled by the processAudio controller function. This function uses a temporary storage mechanism rather than a persistable one:

A computer screen shot of a program code

AI-generated content may be incorrect.

The described methodology allows for the secure audio files to exist in the system only for a limited period. The temp file is deleted immediately after the transcription and summarization processes have been done, even in some unexpected situations due to the final clause. This mitigates unauthorized access to audio recordings while still reducing the risk of data exposure.

## Database Security Procedures for Summaries And Transcriptions

MongoDB stores audio-derived transcriptions and summaries within Mongoose schemas that maintain their integrity and security. The Summary model is designed to maintain direct reference with the user who created it:

A screen shot of a computer program

AI-generated content may be incorrect.

In all controller function executions, this relationship is enforced by explicit permission validations. For instance, when retrieving a summary, the application checks if the user requesting it is indeed the owner:

A screen shot of a computer program

AI-generated content may be incorrect.

In this manner, users are protected in being able to only see their own data even if they obtain a valid ID of another user's summary.

## Account And Password Security

User passwords are never kept in an easy-to-read format (plain text). Instead, user passwords are hashed with Bcrypt encryption which implements a one-way cryptographic function with salt to defend against rainbow table attacks:

A computer screen with text

AI-generated content may be incorrect.

For Google OAuth, no password is stored. Only the Google ID associated with the user's account is safely stored, so there is even less credential exposure:

A screenshot of a computer

AI-generated content may be incorrect.

The “select: false” means that even administrators with the highest levels of access to the system cannot obtain passwords during routine operations which further enhances security.

## Transport Security Protocol Implementation

Eavesdropping and interception attempts on data while being transferred from the client to the server are blocked with modern transport security techniques.

## Implementation of HTTPS

The application enforces HTTPS for every form of communication. The React frontend hosted on AWS Amplify also automatically converts HTTP requests to HTTPS. TLS termination is handled by the platform on the backend devolved on Render, ensuring encryption on all API communications.

## Security of API Requests

Several security measures are included in API requests. The backend server uses a CORS policy that only allows known origins to access the resources:

A computer screen with text

AI-generated content may be incorrect.

For authenticated routes, the application provides an Authorization header with a JWT token:

A computer screen with text

AI-generated content may be incorrect.

This guarantees the resource can be accessed only by authenticated users, and defenses against Request Forgery Attacks are also put in place.

## Implementation of System Authentication

The authentication system implements multiple approaches for convenience and security to the users at the same time.

## Managing Users Sessions Based on JWT Tokens

User sessions are managed with JSON Web Tokens (JWT). These tokens are securely signed using a secret key and contain a valid duration after which they become invalid:

A computer screen with text

AI-generated content may be incorrect.

A token is created and sent to the client whenever a user logs in or registers. This token will be kept inside an HTTP-only cookie to defend against XSS and within localStorage for easy access through the front end:

A computer screen shot of a program code

AI-generated content may be incorrect.

## Protection Middleware

Protected routes include middleware that checks verifications and handles the JWT token on every request.

A screen shot of a computer program

AI-generated content may be incorrect.

Protected resources are blocked off and strcitly guarded. Authentication checks are done per route, shielding routes that need authentication.

## OAuth Integration

Automatic passwordless sign in ability through Google OAuth integration is implemented into the application. In the frontend, the library @react-oauth/google is used.

A screen shot of a computer program

AI-generated content may be incorrect.

After authenticating with Google, this piece of code is responsible for sending the user’s profile details to the backend to either create a new user or update an existing one. That means users do not have to create a password which makes it easier and safer to login.

## Password Reset Implementation

This implemented password reset functionality includes flows that securely prevent account take over.

1. For a user requesting to reset a password, a random token is created.

A computer screen with text and numbers

AI-generated content may be incorrect.

2. This token gets sent to the mentioned email address through a secure email service.

A screen shot of a computer program

AI-generated content may be incorrect.

3. That link permits the user to set a new password if the token check is conducted firstly.

A screen shot of a computer program

AI-generated content may be incorrect.

This implementation allows users claiming registered email to reset easily but ensures that they must be authorized to claim the email to do so.

## Cloud Infrastructure Security

Multitudes of services can be used simultaneously making the application utilize numerous cloud services without compromising on their individually tailored security features designed to improve the overall security infrastructure.

## AWS Amplify Security (Frontend)

The frontend application is hosted on AWS Amplify, which provides several security benefits such as:

* HTTPS configuration automation paired with managed certificates
* AWS Shield's built-in DDoS attack prevention
* Content distribution via CloudFront with edge location cache
* Secure build environments in the CI/CD pipeline

## Render Platform Security (Backend)

The backend API is deployed on Render, which offers:

* Per-service runtime environment isolation
* Automatic management for TLS certificates
* Secure storage for sensitive environment variables
* Regular platform security maintenance

## MongoDB Atlas Security (Database)

The database is hosted on MongoDB Atlas, which includes:

* TLS/SSL protected data in transit
* Data-at-rest encryption for stored data
* Network perimeters through IP allowlisting
* Regular automated secure off-site backups

## AI Security Considerations

The OpenAI integration for transcription and summarization has its own considerations that are addressed within the implementation.

## API Key Security

The OpenAI API key is set as an environment variable on the server, thus not sent to the client:

A black screen with orange text

AI-generated content may be incorrect.

This nullifies the risk of AI service abuse and the possibility of credential exposure.

## Content Security

All the content processed by the AI system undergoes content security measures during its lifecycle. The application guarantees that:

* Users’ audio files undergo processing only a single time and are subsequently deleted
* The resulting transcription is tied to specific user accounts
* Only the owner can access the AI-generated summaries
* No content from any user account is shared with other accounts

## Conclusion

The Meet Summarizer AI application's security framework is sophisticated and multifaceted, guarding data during storage, processing, and transmission. Through careful design of authentication mechanisms, permission validations, and employing robust cloud computing infrastructure, the confidentiality and integrity of information is maintained, and yet user experience is optimized.

As evidenced through provided code samples, the security measures are not hypothetical but have tangible links to the codebase. These implementations adhere to industry standards for web application security and introduce multiple, interdependent protective layers that collectively fortify the entire system.

Traditional methods combined with innovative approaches tailored for Cloud and AI applications deliver a secure environment within Meeting Minutes AI enabling users to transcribe and summarize their meetings at will, without any restrictions on operating the system.