



SHOUT - Audio Management

Daniel Hay

04/07/2025

Course: Software Engineering Stage 6

<https://github.com/Daniel-Hay/AssessmentTask3>
<https://task3-shout-audiotranscription.streamlit.app/>

Table of Contents

1. Identifying and Defining
 2. Research and Planning
 3. System Design
 4. Producing and Implementing
 5. Testing and Evaluation
 6. Client Feedback and Reflection
 7. Appendices
-

1. Identifying and Defining

1.1 Problem Statement

In both educational and professional settings, information is communicated through aural content such as meetings, voice memos and lectures. However, due to the faster and one time nature of spoken communication, individuals struggle with retaining and understanding the information in real time, creating misunderstandings and confusion. This is especially challenging for students and professionals as they need to balance listening and note taking simultaneously. This project aims to assist individuals by providing an audio transcription and customisable summarisation tool, making the process of recording and retaining information easier.

1.2 Project Purpose and Boundaries

The purpose of this project is to allow users to upload their personal audio files, automatically transcribing and summarising the audio into digestible, easy to follow notes. This aims to help individuals, mainly students and professionals, to save their time, make content more accessible, and allow users to focus in the present moment rather than taking down notes. By utilising this transcription and summarisation tool, we reduce the need for users to split their attention into both manual note taking and understanding the content, helping users retain relevant information.

The scope of this project includes the ability to upload pre-recorded audio files that will be transcribed through the Whisper library. Users will then be provided with the opportunity to customise the length of the summary created through a user-friendly web interface. The summarisation will be done through Summa, a keyword extraction tool based off of the TextRank algorithm. Out of scope is real-time text transcription and speaker identification as this would be harder to optimise and be more resource intensive to run and implement.

1.3 Stakeholder Requirements

The stakeholders of this project are the professionals and students that rely on audio content found within lectures, meetings, and voice memos. These users require a program that allows them to convert spoken content into text summaries. Therefore the system needs to accurately transcribe common audio formats such as .mp3 and .wav into accurate, reliable and customisable summary. Additionally, authentication is required to ensure the security of personal details and information stored within the summaries. The ability to save and revisit summaries are integral for repeated use, allowing users to consult their notes at any time. Lastly, an intuitive interface paired with error handling that explains the issues occurring, increases the overall accessibility, enabling users of varying technical knowledge to utilise.

1.4 Functional Requirements

Functional requirements are the core features that the application must integrate in order to achieve the user requirements and goals. These include: a login and registration system that allows users to be uniquely identified and authenticated against a database, the ability to upload supported audio formats, the transcription of audio into text using the Whisper model, the creation of a summary based off a desired length input and the ability to save these summaries, annotating them with a title and optional tags for further identification. Additionally the user must be able to view and delete past summaries. Error handling must be present in all facets of the system to ensure user's understanding of the problem

1.5 Non-Functional Requirements

Non-functional requirements are specifications on how the application should perform, rather than the actual functionality. The program must ensure a secure, reliable and intuitive experience for all users. For security, passwords must be hashed and stored using SHA-256 encryption that prevents personal information from being exposed. Usability is seen through the simplicity of the user interface, that includes helpful and clear instructions to guide users through the software. Performance is another major consideration, the optimisation of loading speeds when transcribing and summarising should be within a reasonable time period to avoid user frustration.

1.6 Constraints

The constraints of this project are the technical limitations of utilising the libraries of Whisper and Summa. Due to the limiting computer power, longer form audio takes a significant amount of time to process, ultimately worsening the user experience. Additionally, as the project is constructed with the free plan of OpenAI's Whisper API, the accuracy of transcription and the support for different languages is limited.

2. Research and Planning

2.1 Development Methodology

The agile methodology is an iterative approach to software development, breaking down the project into smaller, more manageable segments known as sprints. These sprints allow developers to adapt quickly to feedback, granting them the ability to continuously deliver working components that incrementally improve, creating a faster, more efficient environment for success. I chose this methodology as it supports a more flexible approach to development, allowing me to focus on singular features at a time, ultimately ensuring a more organised workflow. Additionally, this structured approach allows for the consolidation and refinement of functions based on particular points of feedback, gradually meeting the goals outlined in the proposal without being overwhelming.

2.2 Tools and Technology

Language: Python 3.9.1

IDE: VSCode

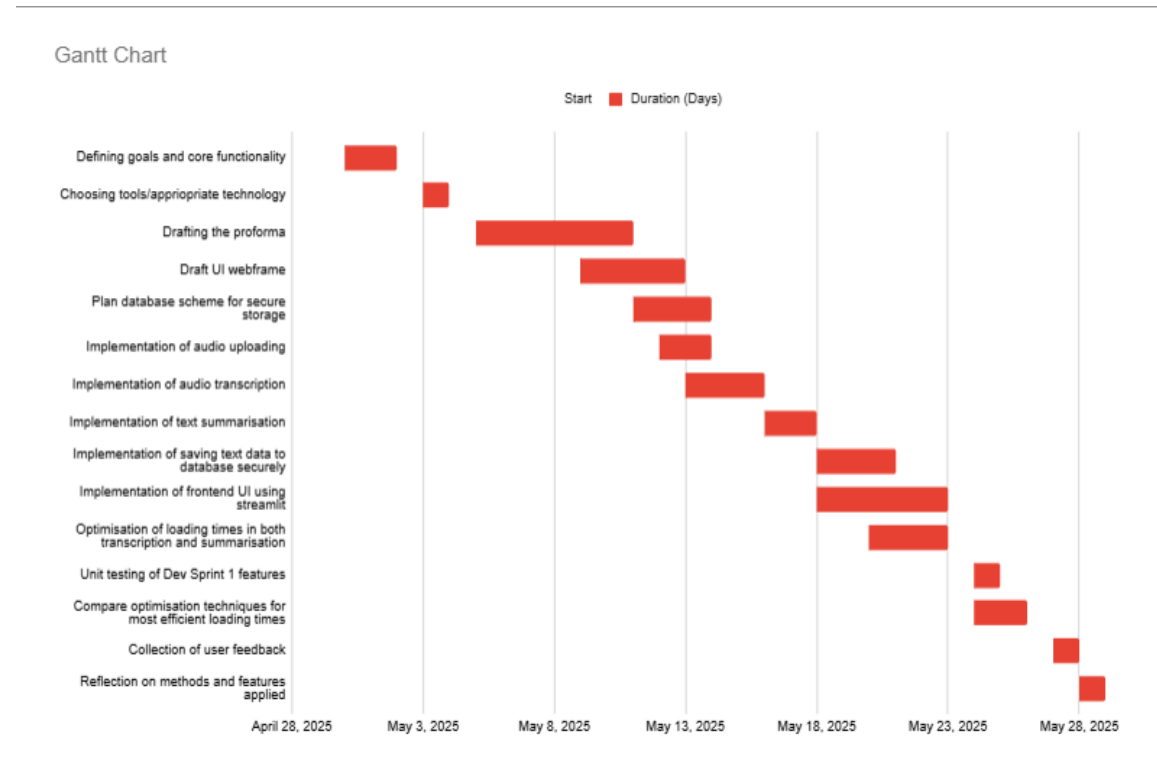
Libraries:

- OpenAI Whisper, StreamLit, Summa, Numpy, Os, Hashlib, SQLite3

Modules:

- Tempfile

2.3 Gantt Chart / Timeline



Planning	4/30/2025	5/2/2025	2 Defining goals and core functionality	2
Planning	5/3/2025	5/4/2025	1 Choosing tools/appropriate technology	1
Planning	5/5/2025	5/11/2025	6 Drafting the proforma	6
Design	5/9/2025	5/13/2025	2 Draft UI webframe	4
Design	5/11/2025	5/14/2025	2 Plan database scheme for secure storage	3
Development Sprint 1	5/12/2025	5/14/2025	3 Implementation of audio uploading	2
Development Sprint 1	5/13/2025	5/16/2025	3 Implementation of audio transcription	3
Development Sprint 1	5/16/2025	5/18/2025	5 Implementation of text summarisation	2
Development Sprint 2	5/18/2025	5/21/2025	6 Implementation of saving text data to database securely	3
Development Sprint 2	5/18/2025	5/23/2025	10 Implementation of frontend UI using streamlit	5
Development Sprint 2	5/20/2025	5/23/2025	5 Optimisation of loading times in both transcription and summarisation	3
Testing	5/24/2025	5/25/2025	4 Unit testing of Dev Sprint 1 features	1
Testing	5/24/2025	5/26/2025	3 Compare optimisation techniques for most efficient loading times	2
Evaluation	5/27/2025	5/28/2025	1 Collection of user feedback	1
Evaluation	5/28/2025	5/29/2025	2 Reflection on methods and features applied	1

Link for better quality: [📅 Gantt Chart - Daniel Hay](#)

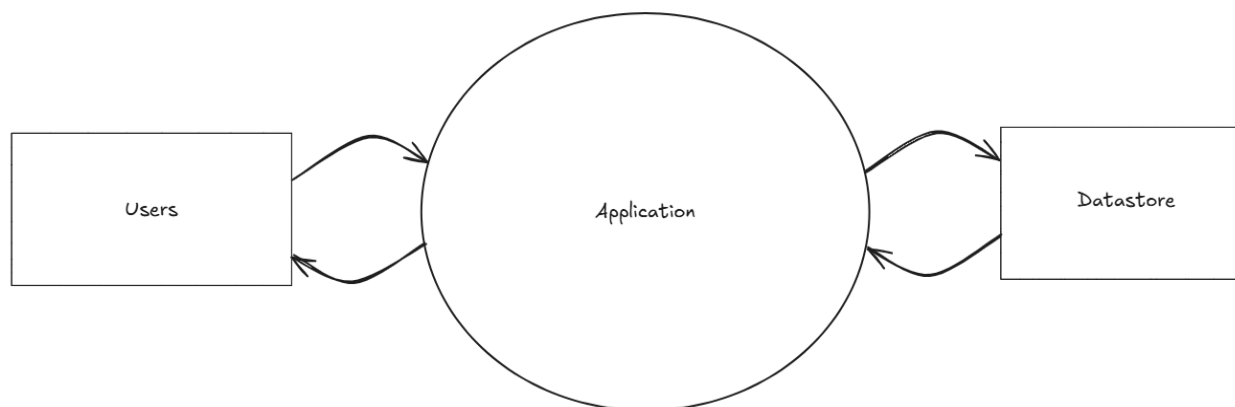
2.4 Communication Plan

To ensure the project meets user requirements, communication within the form of user feedback will be collected and utilised to enhance the overall experience. By gathering information from real world users who represent the target demographic, the program can be improved to better suit their needs. This will be simulated through peer feedback and testing, with each development sprint including a revision process that aims to evaluate the newly created feature, realigning the project to its intended goals.

3. System Design

3.1 Context Diagram

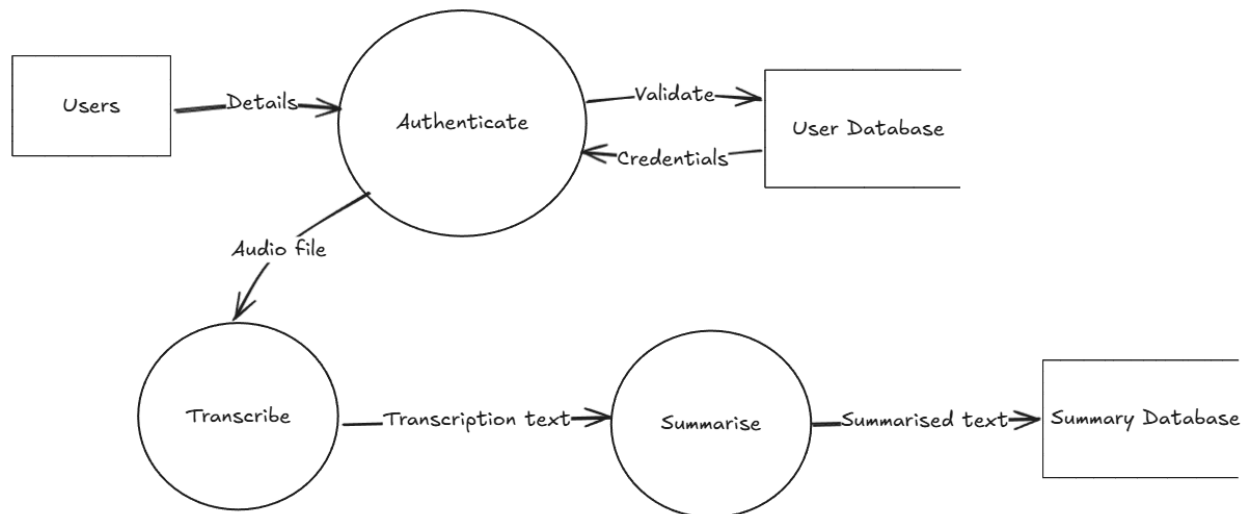
Provides a high level overview of the program, briefly describing the interaction between users, the application and datastore.



3.2 Data Flow Diagrams (1 and 2)

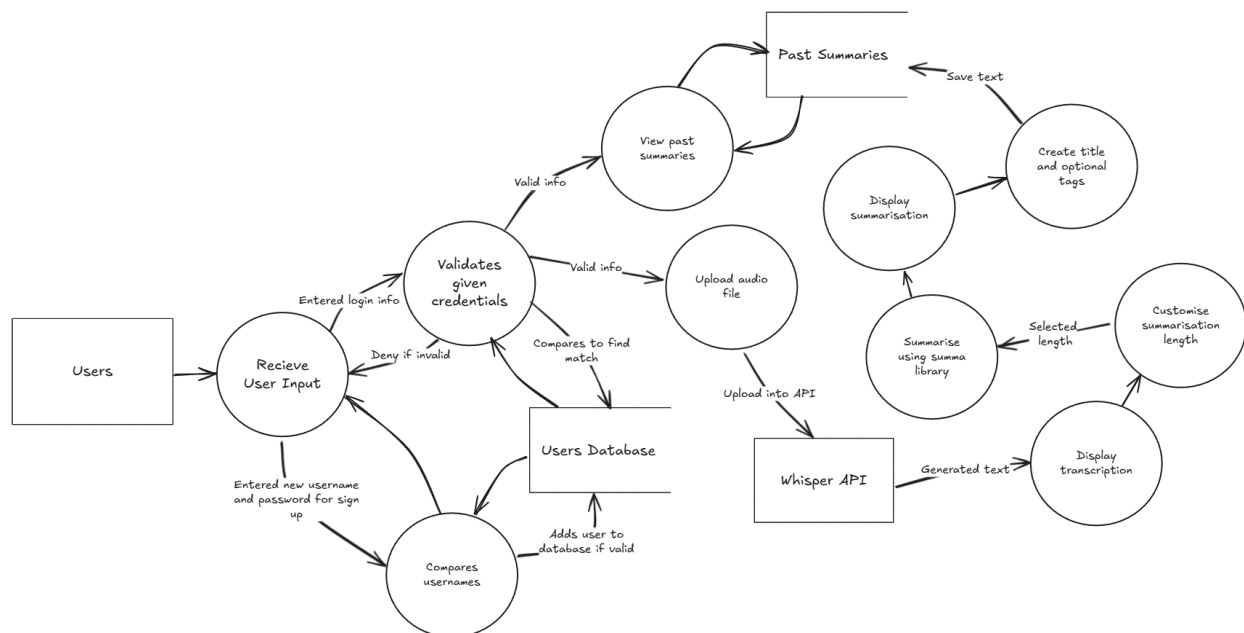
Level 1 Data Flow Diagram

Breaks down the application process into multiple components such as authentication, transcription and summarisation. Outlines the interaction between authentication and the user database in order to validate credentials. Additionally illustrates the simple data flow and hierarchy of processes within the system.



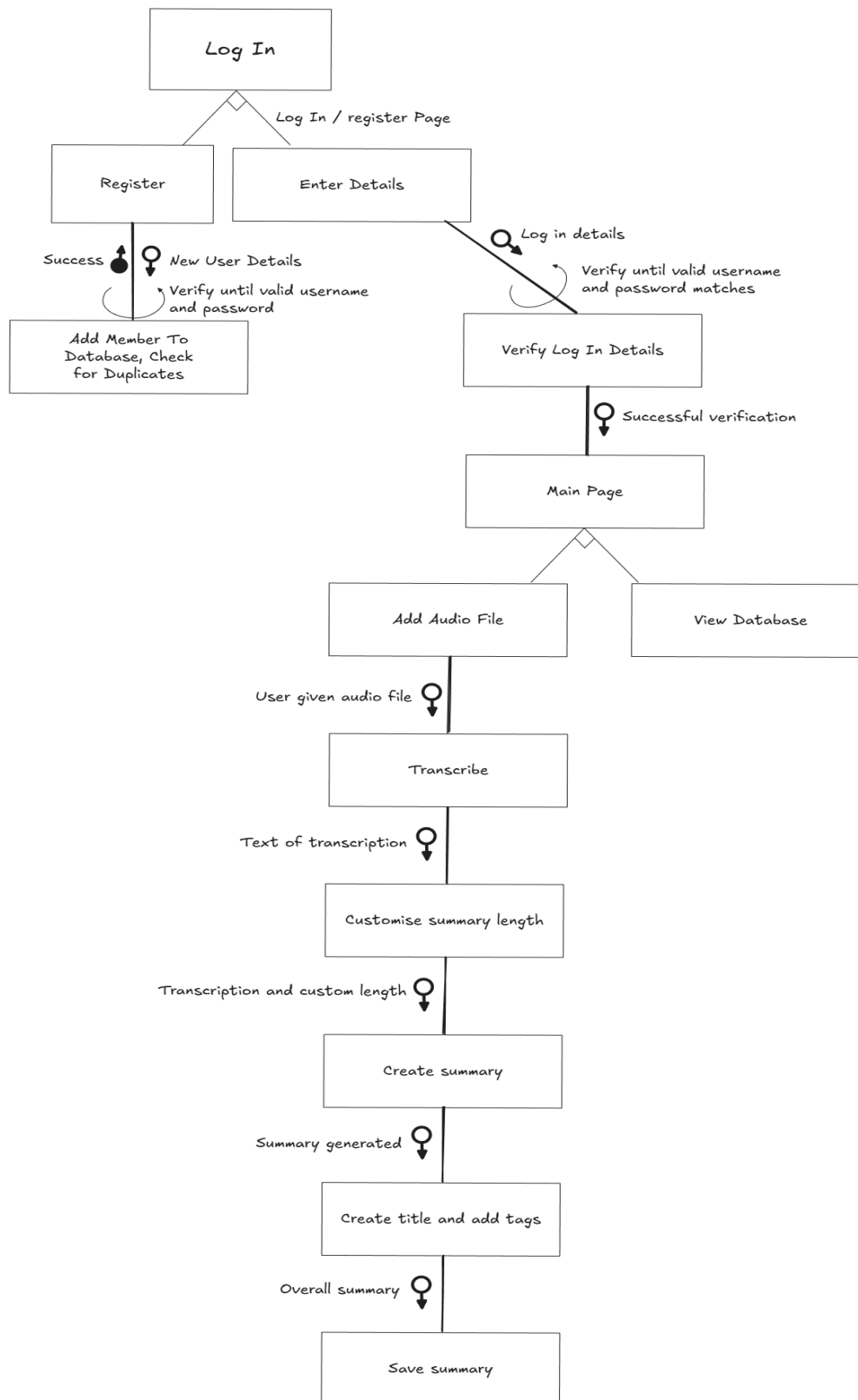
Level 2 Data Flow Diagram:

Breaks the processes down further from a level 1 data flow diagram, creating a deeper understanding of how data flows within these subprocesses and their interactions with data stores and external entities.



3.3 Structure Chart

The structure chart visually illustrates the hierarchical structure of features.



3.4 IPO Chart

Input	Process	Output
Audio file (MP3 and WAV)	<ul style="list-style-type: none">- Check file size and contents using the os module- Transcribe into text using whisper- Summarises using summa	<ul style="list-style-type: none">- Transcribed text- Summarised text
User login credentials	<ul style="list-style-type: none">- Compares to user database to check for duplicates	<ul style="list-style-type: none">- Authorises or prevents access
New user registration info	<ul style="list-style-type: none">- Validate fields- Check for duplicates- Hash password and store user in the database	<ul style="list-style-type: none">- New user account created or error shown
User summarisation preferences	<ul style="list-style-type: none">- Use input to calculate summary ratio- Pass ratio into summarisation function	<ul style="list-style-type: none">- Custom length summary generated
Request to delete a summary	<ul style="list-style-type: none">- Locate and remove selected summary entry from database	<ul style="list-style-type: none">- Updated list of summaries

3.5 Data Dictionary

Variable	Data Type	Format	Description	Example	Validation
Username	String	Alphanumeric	Username of unique customer	DanielHay1	Maximum 15 alphanumeric characters, cannot be blank
Password	String	Hashed Value	User's secret details used for login	PassWord21	Mixture of letters and numbers, must be more than 5 characters and less than 10

AudioTitle	String	Alphanumeric	Users given name for uploaded audio	MathLecture 6	Alphanumeric , can't be null
Transcription	String	Alphanumeric	Text automatically created from given audio file	Today guys we'll be learning about mathematics	Text only, cannot be blank
Tags	String	Alphanumeric	User given information tags for each audio summary	Mathematics Mr Garry Lecture	Text only, cannot be blank
Date	Date and Time	YYYY/MM/DD	Date and time given when user inputs audio file	20/05/2025	Must be a valid date

4. Producing and Implementing

4.1 Development Process

How did you approach the building of your solution?

The building of the project was done with the AGILE methodology of short, iterative sprints that break tasks down into manageable parts. In order to ensure a solution that meets the requirements of stakeholders, I first designed an initial proposal form that outlines the key elements of my designed solution, allowing stakeholders to review and provide feedback before committing resources to the task. Additionally, the creation of diagrams such as data flow diagrams and structure charts before beginning the development process allows me to better visualise data flow and the overall hierarchy of functions and how they interact. The first feature developed within these sprints was the core feature of transcription and summarisation, before additional features such as the login system and saving text functions. By only focusing on one feature at a time, it allowed me to isolate any problems or errors that arose, improving overall efficiency of development.

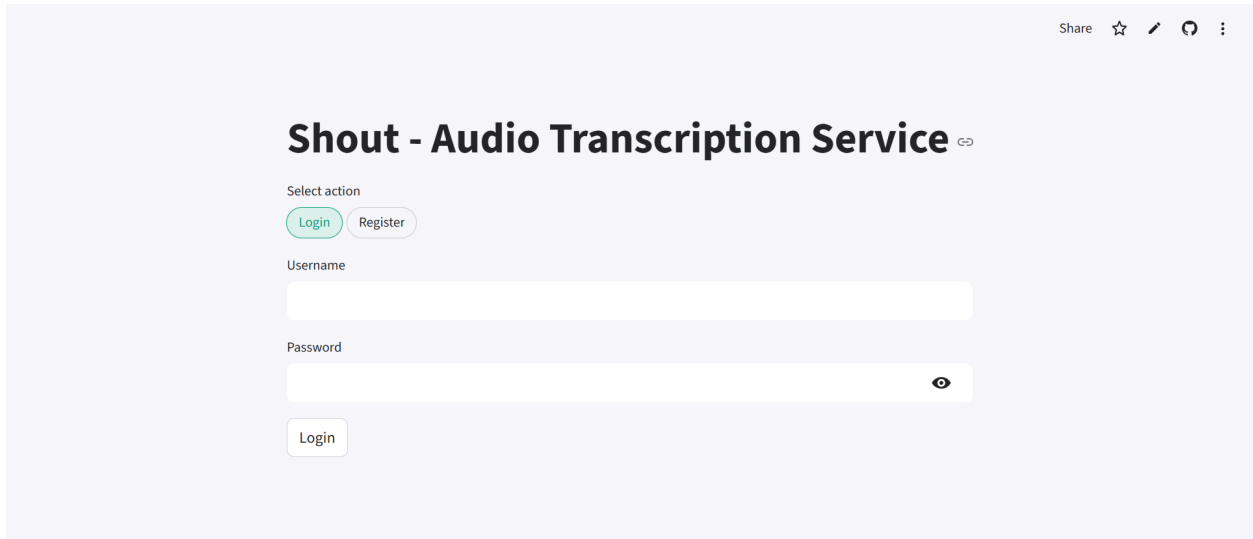
4.2 Key Features Developed

Feature	Description and Justification
User Registration	Allows new users to create an account. Usernames and passwords are stored securely using SHA-256 hashing. This ensures that user data is protected and personalisation is possible.
User Login	Authenticates users by comparing their credentials against the database. This is utilised to ensure the security of the application, only allowing authorised users to access personal data and information.
Audio upload	Enables users to upload mp3 and wav files for future transcription. This is necessary as it provides the raw data required for transcription and later summarisation.
Transcription	Converts audio files into text using the OpenAI Whisper API. This is a core function of the project, as transcription is required to translate audio into text for summarisation can occur.
Adjustable summary length	Allows users to adjust summarisation length to fit their requirements. This is done to provide flexibility, being able to be manipulated to fit changing user requirements.
Summary generation	Uses Summa library that creates summarised sentences based on the Textrank algorithm, making larger amounts of text more understandable in shorter periods of time.
Save summary	Stores the user's summarised notes in an SQLite database along with tags and timestamps. This enables users to revisit and organise important information over time.
View summarises	Allows users to browse and read all their previously saved summaries. This supports study revision and ongoing access.
Error handling	Detects issues like unsupported audio formats, empty files, or failed logins. Provides clear, user-friendly error messages to improve the user experience.

4.3 Screenshots of Interface

Login Menu:

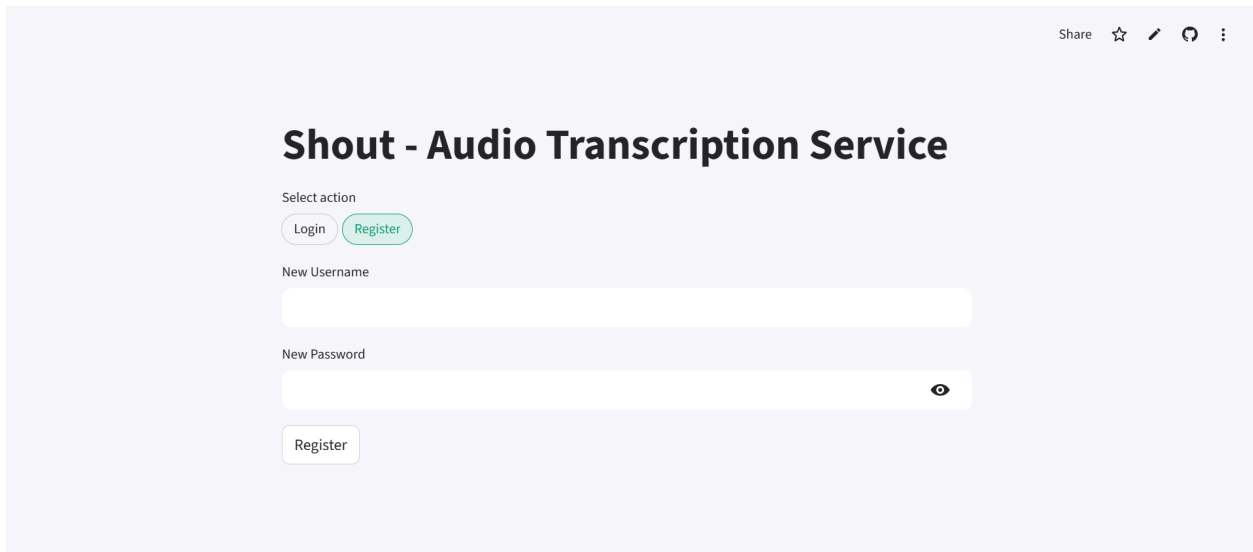
Allows users to access a previously made account. Enables users to enter usernames and passwords which are compared against stored credentials. This ensures only authorised users can access personalised features such as transcription and saved summaries.



The screenshot shows the login interface for 'Shout - Audio Transcription Service'. At the top right, there are icons for 'Share', a star, a pencil, a refresh, and a menu. The title 'Shout - Audio Transcription Service' is centered, followed by a link icon. Below the title, there is a 'Select action' section with two buttons: 'Login' (highlighted in green) and 'Register'. Underneath, there are input fields for 'Username' and 'Password'. The 'Password' field has a toggle icon (an eye) to the right. At the bottom, there is a 'Login' button.

Registration Menu:

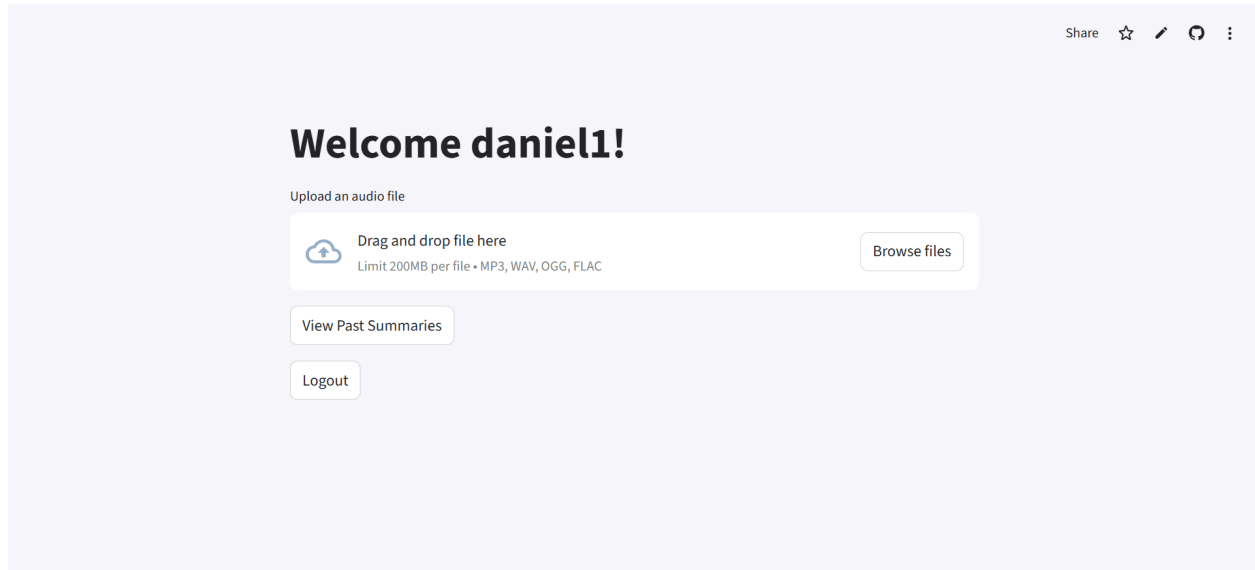
Allows users to create new accounts, enabling them to enter a unique username and password for future use. By hashing the passwords created, their information is securely stored, preventing the potential for malicious attacks.



The screenshot shows the registration interface for 'Shout - Audio Transcription Service'. At the top right, there are icons for 'Share', a star, a pencil, a refresh, and a menu. The title 'Shout - Audio Transcription Service' is centered. Below the title, there is a 'Select action' section with two buttons: 'Login' and 'Register' (highlighted in green). Underneath, there are input fields for 'New Username' and 'New Password'. The 'New Password' field has a toggle icon (an eye) to the right. At the bottom, there is a 'Register' button.

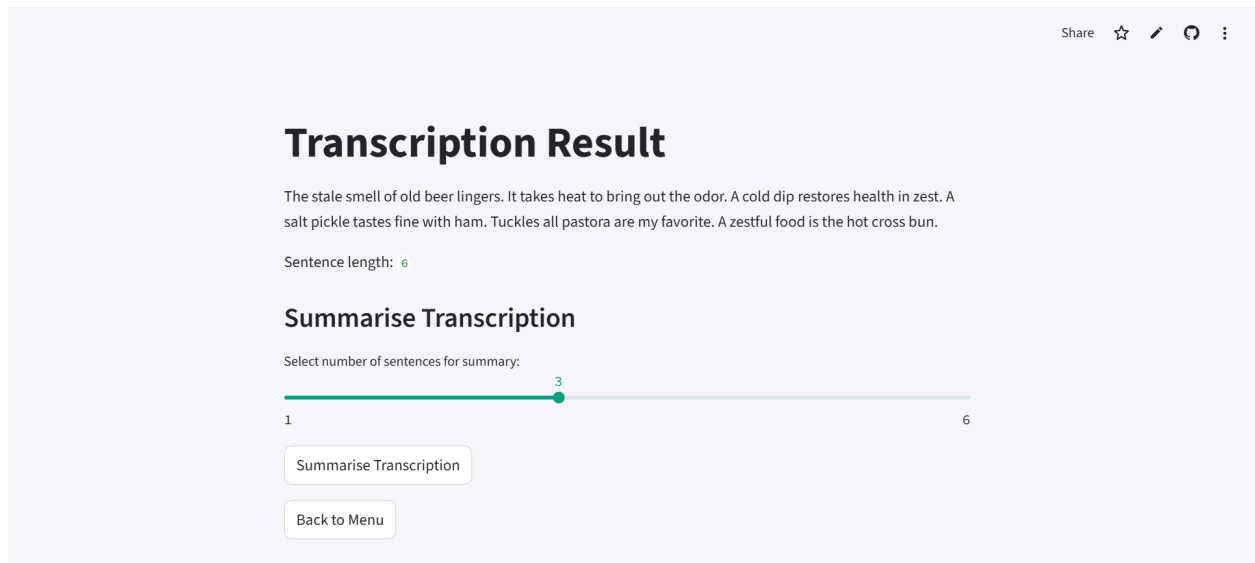
Main Menu:

Allows users to navigate between submitting audio files for transcription and their past summaries. Enables browsing or drag and drop of audio files for transcription through an intuitive interface and widget.



Transcription Page and Customisation Widget:

Allows users to view transcription and select how many sentences to summarise into via a slider.



Saving Menu:

Input fields to allow users to enter information such as a title and tags for the user to easily find their summaries later.

The screenshot shows a web interface titled "Summarise Transcription". In the top right corner, there are icons for "Share", a star, a pencil, a circular arrow, and a three-dot menu. Below the title is a light blue box containing a notepad icon and the text "Summary:". Underneath this box, a sample summary is displayed: "A cold dip restores health in zest. A zestful food is the hot cross bun." Below the sample text is a green button labeled "Save Summary".

Save Your Summary

Title for your summary

Tags (comma-separated)

Submit

Back to Menu

Summarisation Viewer:

Allows users to easily navigate through past summaries via titles, tags or dates. Additionally gives the option to delete past summaries.

The screenshot shows a web interface titled "Your Saved Summaries". In the top right corner, there are icons for "Share", a star, a pencil, a circular arrow, and a three-dot menu. Below the title, a specific summary is displayed with the title "Test123". Underneath the title, the tags and save date are shown: "Tags: tagtest | Saved: 2025-07-03 11:48:14". The summary text is "Daniel is a funny, funny person. It's taken some time to go onto it." Below the text is a button labeled "Delete Summary".

Back to Menu

4.4 Version Control Summary

Fig. 4.4 - Evidence of ongoing commits and progress (see github)

Generally speaking I commit code 5 times a week. This can be seen in the activity tracker online. My commitments and changes increased over time, with the majority of small changes happening towards the end of the project, fixing minor bugs and issues with deployment.

Code frequency over the history of Daniel-Hay/AssessmentTask3



Commits on Jun 2, 2025		
Added hashing functionality within the passwords	Daniel-Hay committed on Jun 2	3cccdad
Added User Registration and Error Handling	Daniel-Hay committed on Jun 2	ac45ff4
Commits on May 30, 2025		
DOCs	Daniel-Hay authored on May 30	53a3449
Commits on May 26, 2025		
Added summarisation feature	Daniel-Hay committed on May 26	d0d3e2c
Commits on May 16, 2025		
Initial upload for cloning	Daniel-Hay authored on May 16	bfcde8b
Commits on May 5, 2025		
Initial commit	Daniel-Hay authored on May 5	91fe965

	Creation of saving system <small>new</small> Daniel-Hay committed last week	2bfaa08		
Commits on Jun 17, 2025				
	Changed UI elements, more modern aesthetic feel Daniel-Hay committed 2 weeks ago	ba49c24		
	Debugged and fixed errors regarding transcription and summarisation <small>new</small> Daniel-Hay committed 2 weeks ago	4f1f517		
Commits on Jun 5, 2025				
	Fixing errors within summarisation <small>new</small> Daniel-Hay committed last month	6959d46		
Commits on Jun 3, 2025				
	Optimise loading speed for transcription <small>new</small> Daniel-Hay committed last month	aba9e0d		
	Fixed some errors within reading audio files <small>new</small> Daniel-Hay committed last month	88dd6cb		
	Reworked login system <small>new</small> Daniel-Hay committed last month	986bfc7		

	Added testing Daniel-Hay committed 11 hours ago	2203de5		
Commits on Jun 30, 2025				
	Attempt to fix audio reshaping errors Daniel-Hay committed 3 days ago	9d4aada		
	Fixed ffmpeg error Daniel-Hay committed 3 days ago	75d711f		
	Added requirements Daniel-Hay committed 3 days ago	fdcfsf2		
	Created a requirements file for deployment Daniel-Hay committed 3 days ago	afe7b7a		
Commits on Jun 28, 2025				
	Created new python file for unittesting Daniel-Hay committed 5 days ago	9a6ebd7		
	Add delete summary feature to review page Daniel-Hay committed 5 days ago	0411fd7		
	Displayed the user summarises data from the table into the interface Daniel-Hay committed 5 days ago	ce627eb		

5. Testing and Evaluation

Testing is the process of evaluating and verifying a component or application meets the requirements as expected, ensuring that the program performs reliably, handles errors correctly, and delivers accurate results to the user. Following the AGILE method, testing was done after every sprint, allowing for frequent feedback for subsequent fixing.

5.1 Testing Methods Used

User Testing: Individuals interact with the program to provide feedback and identify areas for improvement

Unit testing: Individual components are tested in isolation, ensuring that each unit works correctly in isolation before being integrated into the larger system.

Integration testing: The addition of multiple components and tests the functionality of them. Ensures that the interaction between parts properly works.

Acceptance testing: Final test performed by end users to formally verify if a system meets the user's needs and requirements

5.2 Test Cases and Results

Test	Type	Description	P/F
Hashing of password	Unit testing	<p>Tested whether the password is properly hashed and stored within the SQLite database using SHA-256 rather than plain text. Done in an external file.</p> <pre>def test_hash_password(self): password = "asdfasdf" hashed = hash_password(password) self.assertNotEqual(hashed, password)</pre>	P
Registration	Unit testing	<p>Verifies that adding a user actually inserts that user into your database. This allows for new users to register and be identified uniquely.</p> <pre>def test_add_user(self): add_user("warren", "asdfasdf") c.execute('SELECT * FROM userstable WHERE username=?', ("warren",)) user = c.fetchone() self.assertIsNotNone(user)</pre>	P
Login system	Unit testing	<p>Verifies that users can log in with the correct credentials if they match up with the database.</p> <pre>def test_login_user(self): add_user("warren", "asdfasdf") user = login_user("warren", "asdfasdf") self.assertIsNotNone(user)</pre>	P
Saving system	Unit testing	<p>Ensures that a generated summary is saved to the database with a title and optional tags.</p>	P

		<pre>def test_save_summary(self): add_user("warren", "asdfsdf") save_summary("warren", "Test Title", "Hello, my name is warren and I like to party", "tag1,tag2") c.execute('SELECT * FROM summaries WHERE username=?', ("warren",)) summary = c.fetchone() self.assertIsNotNone(summary)</pre>	
Transcription accuracy	User testing	Ensures that the core transcription feature performs reliably in realistic conditions with user provided audio files. Users rated how accurate the Whisper model was to the actual content.	P
Error handling	User testing	Verifies that the application correctly detects and responds to invalid user actions such as providing corrupted audio files or incorrect login information. Done through allowing users to explore the application without instructions.	P
Whole application	Integration testing	Confirms that the features work in tandem with each other without any issues arising. Done through users going through the typical flow of logging in and creating a transcription.	P
Final version	Acceptance testing	Verification on whether the combined features meet the end goal of helping individuals be more present in the moment by transcribing audio into summarised notes that can be recalled for future revision. Done by end users in real world scenarios.	P

5.3 Evaluation Against Requirements

The software solution meets the original requirements set out in the proposal as it provides a functional, accurate and user friendly tool that transcribes and summarises audio content. All functional and non-functional requirements were completed and integrated into the program. Furthermore, user testing and feedback after development sprints demonstrate the effectiveness of the program in meeting the user requirements in real world applications, successfully allowing individuals to stay present during lectures or meetings without worrying about missing important information.

5.4 Improvements and Future Work

With extra time, I would have implemented real-time audio transcription and an inbuilt text editor to fix any slight mistakes. Real time transcriptions provide more immediate useful information whilst the text editor creates more flexibility for users to generate a more accurate transcription.

6. (Client) Feedback and Reflection

6.1 Summary of Client Feedback

List of client comments or peer feedback.

Peers	Comments/Feedback
1.	The application is very easily navigable, and allows for very quick and simple translation from audio to text. The summarisation function is very convenient, providing me with the option to understand the information under time pressured situations.
2.	This app is powerfully accessessible, and enables me for premature and basic translation from sounds to words. The summarisation function is very efficient, granting me the potential to comprehend the data during a certain time period when placed under tense scenarios.
3.	Add a feature that utilises AI to suggest edits for summarised scripts, would be useful for people trying to create speeches.
4.	The optimisation speeds could be faster for longer forms of audio files. However, nice job overall, with a clean and modern user interface.

6.2 Personal Reflection

Throughout the creation of my project, I learnt and improved upon both technical and design skills including:

- The real life application of AGILE methodology
- Utilisation of python libraries such as streamlit, sqlite3, whisper and summa
- Secure database design and management using SHA-256 hashing
- How to create different diagrams such as level 2 DFD and structure charts and their real applications
- The design of user interfaces
- How to document a project and create a report

7. Appendices

Links:

Github: <https://github.com/Daniel-Hay/AssessmentTask3>

App: <https://task3-shout-audiotranscription.streamlit.app/>

Code Snippets: (See Below)

```

# Cache model loading, improving performance
@st.cache_resource
def get_whisper_model():
    return whisper.load_model("tiny")

conn = sqlite3.connect('data.db')
c = conn.cursor()
c.execute('CREATE TABLE IF NOT EXISTS userstable (username TEXT, password TEXT)')
c.execute('CREATE TABLE IF NOT EXISTS summaries (username TEXT, title TEXT, summary TEXT, tags TEXT, date TEXT)')
conn.commit()

def add_user(username, password):
    hashed = hash_password(password)
    c.execute('INSERT INTO userstable(username, password) VALUES (?, ?)', (username, hashed))
    conn.commit()

def login_user(username, password):
    hashed = hash_password(password)
    c.execute('SELECT * FROM userstable WHERE username=? AND password=?', (username, hashed))
    return c.fetchone()

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def register():
    new_username = st.text_input("New Username")
    new_password = st.text_input("New Password", type="password")
    if st.button("Register"):
        # Check if username already exists
        c.execute('SELECT * FROM userstable WHERE username=?', (new_username,))
        if c.fetchone():
            st.error("Username already exists. Please choose another.")
        elif not new_username or not new_password:
            st.error("Username and password cannot be empty.")
        else:
            add_user(new_username, new_password)
            st.success("Registration successful! You can now log in.")

```

```

def transcribe_page():
    st.title("Transcription Result")

    uploaded_file = st.session_state.get("audio_file", None)
    temp_audio_path = st.session_state.get("temp_audio_path", None)
    result = st.session_state.get("result", None)

    # Only create temp file and transcribe if not already done
    if uploaded_file and (not temp_audio_path or not os.path.exists(temp_audio_path) or not result):
        file_extension = uploaded_file.name.split(".")[-1]
        with tempfile.NamedTemporaryFile(delete=False, suffix=f".{file_extension}") as tmp_file:
            uploaded_file.seek(0)
            tmp_file.write(uploaded_file.read())
            temp_audio_path = tmp_file.name
            st.session_state["temp_audio_path"] = temp_audio_path

        model = get_whisper_model()
        try:
            audio = whisper.load_audio(temp_audio_path)
            if audio is None or (isinstance(audio, np.ndarray) and audio.size == 0):
                st.error("The uploaded audio file is empty or invalid. Please upload a valid audio file.")
                os.remove(temp_audio_path)
                st.session_state["temp_audio_path"] = None
                return
            result = model.transcribe(temp_audio_path)
            st.session_state["result"] = result
        except Exception as e:
            st.error(f"Failed to transcribe audio: {e}")
            if temp_audio_path and os.path.exists(temp_audio_path):
                os.remove(temp_audio_path)
            st.session_state["temp_audio_path"] = None
    return result

```

```

def main_menu():
    st.title(f"Welcome {st.session_state['username']}!")
    uploaded_file = st.file_uploader("Upload an audio file", type=["mp3", "wav", "ogg", "flac"])

    if uploaded_file:
        st.audio(uploaded_file)
        st.session_state["audio_file"] = uploaded_file

        if st.button("Transcribe"):
            st.session_state["page"] = "transcribe"
            st.rerun()

    if st.button("View Past Summaries"):
        st.session_state["page"] = "review"
        st.rerun()

    if st.button("Logout"):
        st.session_state.clear()
        st.rerun()

```