טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)

הגשת פרויקט

SoundScribe, B17, Team 18

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
|  |  |
|  |  |
|  |  |
|  |  |

SoundScribe is our audio transcription web app, dedicated to converting podcast episodes and audio files of varying sizes to text. The app provides a transcription of the audio, a summary of the topic, timestamps for sentences and a search feature that highlights matching words. Users can upload audio files and get them transcribed through in one of three languages: English, Hebrew, and Arabic. Users can create accounts that will store their transcription data and view it in their Profile page. We’ve included a dark/light theme toggle and made sure the application stays responsive on desktop as well as mobile devices. To monitor user’s experience, we’ve created a star rating system that stores the user’s feedback in our database.

**Frontend** – Mainly React for user interface and components. We used React Router for page navigation and Tailwind CSS for styling and responsiveness. Context API is also used to add language preferences to all our interfaces.

**Backend** – The project relies on AssemblyAI for the transcription and processing of audio files. To manage users we rely on Firebase Authentication services and their data (credentials, transcription and rating) is stored in Firebase Firestore.

Github repo: <https://github.com/muffindonor/web-podcast-transcript>

Soundscribe link: <https://web-podcast-transcript-oind.vercel.app/>

Gitpages: <https://muffindonor.github.io/web-podcast-transcript/>

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| עבד אל חמיד עמאר | הוספת אפשרות לתמלל אודיו בעברית וערבית | הוספת אפשרות לתמלל אודיו בעברית וערבית |
| דני לבובסקי | ביצוע QA, תיקון רספונסיביות של Navbar | ביצוע QA, תיקון רספונסיביות של Navbar |
| שי ליברמן (מהנדס מערכת) | יצירת עמוד פורפיל מתפקד | יצירת עמוד פורפיל מתפקד |
| שהד שלבי | כתיבת תיק מתכנת | כתיבת תיק מתכנת |
| היבה אבו קנדיל | כתיבת תיק משתמש | כתיבת תיק משתמש |

**2.Functional Requirements:**

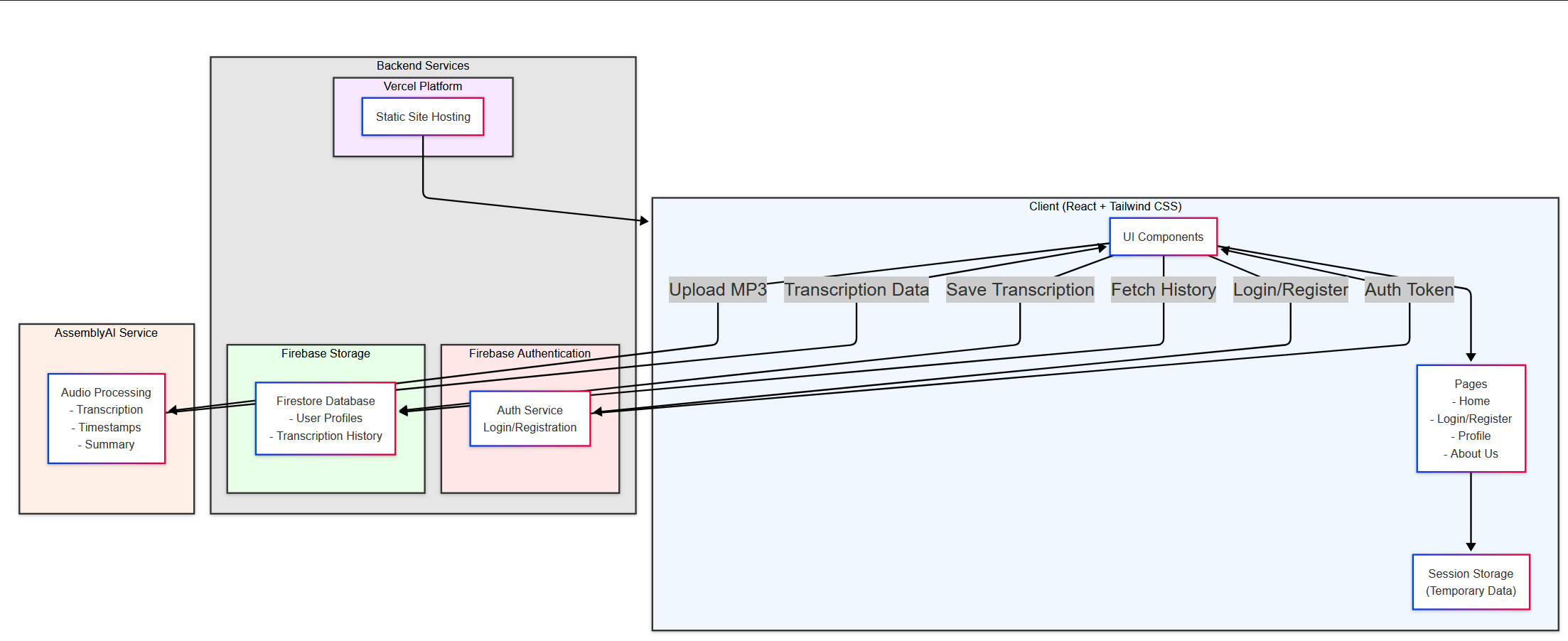
1. Audio upload – Users can upload audio files (MP3, WAV, etc.) to the website.
2. File check - The website will validate the file and provide appropriate error messages for unsupported formats.
3. Transcribe audio – The website will automatically transcribe the uploaded audio into text using speech-to-text technology.
4. Catalog text - The website will automatically create a timestamps for sentences based on the generated text.
5. Summarization - The website will create a summary of the processed text.
6. Rating - The website will allow users to rate their experience.
7. The website will allow users to register and will maintain a user database.
8. The system will allow users to login and view their profile pages.
9. Users will be able to search for keywords and sentences in the transcription.

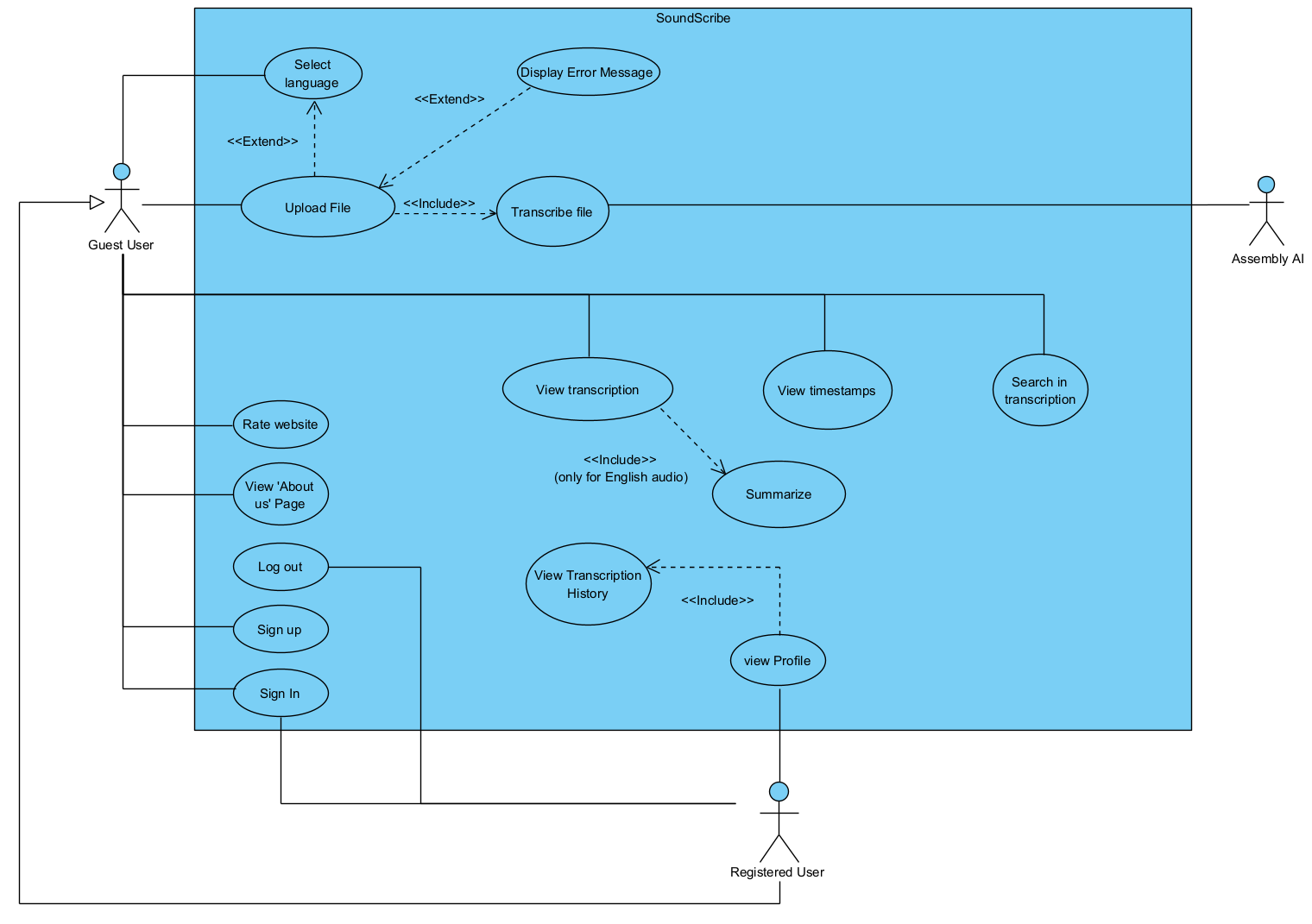
**2.Non-Functional Requirements:**

1. Performance – The system will be able to process and return an audio file’s transcription in under two minutes for files up to size 30MB
2. Scalability – The system will support scalability by improving the database to handle more data, the website deployed to vercel to handle more traffic or by subscribing to a higher tier of AssemblyAI to improve performance
3. SystemAvailability – The system will be available for use for 99.99% of the time and will recover from server crash in, at most, 5 minutes.
4. Accessibility – Sitewide navigation will be simple, the interface will be clear and understandable, pages will adjust to different monitor sizes as needed, users will be able to turn on dark mode.
5. Maintainability – The project’s codebase will be modular and separated into React components allowing future modification or extension of the project’s functionality to be done easily.

**3.מבנה ואינטראקציה**

(לפרויקט מצורפים קבצי הדיאגרמות ברזולוציות יותר נוחות לצפייה)

תרשים ארכיטקטורה:

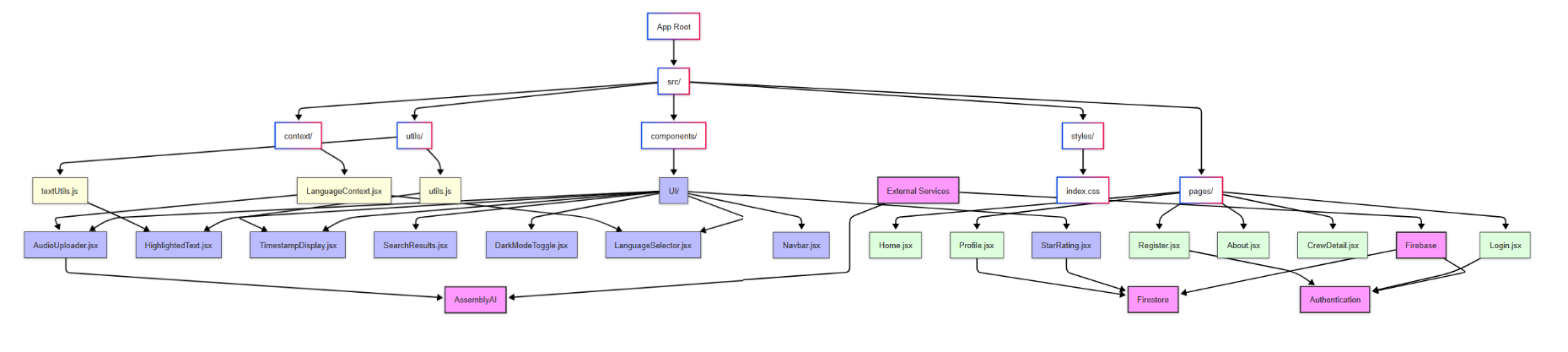
תרשים Usecase:

**4. מבנה סופי של האתר**

(רזולוציית הדיאגרמות נמוכה בעקבות השימוש בוורד, לנחויותכם מצורפים קבצי הדיאגרמות לצד ההגשה)

דיאגרמה לתיאור תיקיות, קבצים וקומפננטות (File&Components.png)

הסבר לכל קומפוננטה נמצא בתיק מתכנת תחת UI Components, Page Components



דיאגרמה לתיאור מסד הנתונים (database.png)

A screenshot of a diagram

Description automatically generated

# 5. תיק מתכנת:



SoundScribe is our audio transcription web app, dedicated to transcribing podcast episodes but can handle audio files of varying sizes. Users can upload audio files and get them transcribed through AssemblyAI. The app supports English, Hebrew, and Arabic, handles user accounts, and includes features like search with highlighting and user ratings. We built it using React, Firebase for authentication and storage, and Tailwind CSS for styling and responsiveness.

# Overview of the Application

**Purpose & Features:**

* **Upload Audio Files:** Files are sent to AssemblyAI for transcription.
* **Receive Transcriptions:** Users get the full text, summarization (for English only), and timestamps for sentences and words.
* **Search & Highlight:** Users can search the transcribed text. The results are then highlighted in the transcription and are presented along with their accurate timestamps.
* **User managing system:** Users can register, log in, and see their transcription history in their profile page. This functionality is operated by Firebase Authentication and Firestore.
* **Rate Your Experience:** We implemented a star-rating component to collect user feedback and store it in the database.
* **Multi-Language Support:** Soundscribe supports English, Hebrew, and Arabic with corresponding layout adjustments.
* **Theme Toggle:** A dark mode toggle lets users switch between light and dark themes.
* **Team/About Pages:** The app includes pages describing the company and the team, with detailed member profiles.

**Framework & Architecture:**

* **React:** For building UI components.
* **React Router:** For routing between pages.
* **Firebase:** For user authentication and database (Firestore) storage.
* **AssemblyAI:** For processing audio uploads and returning transcriptions.
* **Tailwind CSS:** For styling along with custom CSS variables for theming.

# Component and File Breakdown

## **1.** UI Components

**AudioUploader Component**

* **Purpose:**  
  Lets users upload an audio file and initiates the transcription process via AssemblyAI.
* **Key Functions:**
  + **handleFileChange(e)**
    - Validates that the file uploaded is an audio file.
    - Updates local state and clears errors.
  + **getTranscriptionBody(language, uploadResult)**
    - Constructs a JSON payload for the transcription request.
    - Uses different configurations based on the selected language (mainly enables summarization for English).
  + **handleTranscribe()**
    - Handles the transcription:
      1. **Upload:** Send the file to AssemblyAI.
      2. **Transcription Request:** Submits a transcription request.
      3. **Completion:** On success, calls a callback (onTranscriptionComplete) with the transcription data.
  + **UI Elements:**
    - File drop zone with dynamic styling.
    - A button to start transcription (disabled during processing).
    - A progress indicator and error messages.

**HighlightedText Component**

* **Purpose:**  
  Highlights text based on a search query.
* **How It Works:**
  + Calls a utility function (highlightText) that splits the text by the search term and wraps matches in <mark> tags.
  + Uses dangerouslySetInnerHTML to render the processed HTML safely

**TimestampDisplay Component**

* **Purpose:**  
  Displays the transcribed text grouped into sentences together with timestamps.
* **How It Works:**
  + **Sentence Grouping:**  
    Uses the groupWordsIntoSentences utility to convert an array of words (each with timing) into sentence blocks.
  + **Time Formatting:**  
    Each sentence’s start time is converted to MM:SS format (via formatTime).
  + **Display:**  
    Renders a list of sentence items with the formatted timestamp and the sentence text (with search highlights if available).

**SearchResults Component**

* **Purpose:**  
  Searches the transcription for matching segments and displays them with timestamps and context.
* **Key Function:**
  + **findMatchingSegments()**
    - Splits the search query into words.
    - Iterates through the transcript’s words, finding segments that match the query.
    - Includes a few words of context before and after each match.
  + **Display:**  
    For each match, shows a clock icon with the time and the context text with highlighted search terms.

**DarkModeToggle Component**

* **Purpose:**  
  Provides a button that toggles between dark and light themes.
* **How It Works:**
  + **State Initialization:**  
    Reads from localStorage or uses the system’s color scheme to set the initial theme.
  + **useEffect Hook:**
    - Updates the HTML’s data-theme attribute on state changes.
    - Maintains the current theme in user’s localStorage.
  + **UI:**
    - Displays a Sun icon when dark mode is active (to indicate switching option to light) and a Moon icon when in light mode.

**LanguageSelector Component**

* **Purpose:**  
  Allows users to choose between English, Hebrew, and Arabic.
* **How It Works:**
  + Defines an array of language objects (each with a code, label, and flag).
  + Uses the LanguageContext to get and update the current language.
  + Renders buttons that visually indicate the active language.

**Navbar Component**

* **Purpose:**  
  Allows for site-wide navigation and switching themes.
* **Key Features:**
  + **User buttons:**  
    Shows different buttons depending on whether the user is logged in (“Login” and “Register”. If user is logged in then “Profile” and “Logout”).
  + **Mobile Responsiveness:**
    - Contains a mobile menu toggle (hamburger and close icons).
    - Displays an overlay when the mobile menu is active.
  + **Authentication Integration:**  
    Listens to Firebase auth state changes and manages logout.

**StarRating Component**

* **Purpose:**  
  Lets users rate their experience using a 5-star system.
* **How It Works:**
  + **Rating & Hover States:**  
    Uses state to track the selected rating and the current hover value.
  + **Prevent voting fraud:**  
    Checks session storage so that a user can’t rate multiple times in the same session.
  + **Firestore Integration:**  
    Saves user’s rating, along with user’s information and a timestamp, to Firestore.
  + **UI:**
    - Provides clickable star icons with hover effects and a thank-you message after rating.

## 2. Page Components

**Register.jsx**

* **Purpose:**  
  Provides a registration form for new users.
* **How It Works:**
  + **handleRegister(e)**
    - Prevents empty form submission.
    - Uses Firebase’s createUserWithEmailAndPassword to create a new account.
    - Creates a Firestore document for the new user (storing email, creation date, and an empty transcriptions array).
    - Redirects to the home page upon success.
  + **Additional Features:**
    - Password visibility toggle using icons.
    - Error handling to display registeration errors.

**Login.jsx**

* **Purpose:**  
  Offers a login form for existing users.
* **How It Works:**
  + **handleLogin(e)**
    - Uses Firebase’s signInWithEmailAndPassword for authentication.
    - Displays an error if login fails.
    - Redirects to the home page on successful login.
  + **Features:**
    - Like registration page, login page includes a password visibility icon.

**Home.jsx**

* **Purpose:**  
  This is the main page where users upload audio files and see transcription results.
* **How It Works:**
  + **Audio Upload & Transcription:**
    - Handles the AudioUploader component.
    - Passes a callback (handleTranscriptionComplete) to update the state with the transcription result.
  + **Displaying Results:**
    - Shows transcribed text, summary (if English), timestamps (via TimestampDisplay), and a search interface (with SearchResults and HighlightedText).
  + **Save Transcription results:**
    - If a user is logged in, saves transcription data to Firestore (under the user’s document ).
  + **User Interaction:**
    - Provides a search function that updates the displayed text with be highlighted based on user queries.
    - Handles the StarRating component at the bottom for feedback.

**Profile.jsx**

* **Purpose:**  
  Displays the user’s transcription history, accessible only if the the user is logged in.
* **How It Works:**
  + **Authentication & Data Fetching:**
    - Listens for auth state change and redirect to login if the user tries to access the page without being logged in.
    - Retrieves the user's transcriptions from Firestore and displays it.
  + **Sidebar:**
    - Lists saved transcriptions with titles and formatted timestamps.
    - Allows the user to select a transcription to view its details.
  + **Detail View:**
    - Displays the full transcription text, summary, sentence timestamps (via TimestampDisplay), and a search interface like in the main page.

**CrewDetail.jsx**

* **Purpose:**  
  Presents detailed information for each individual team member.
* **How It Works:**
  + Uses URL parameters (via useParams) to determine which team member to display.
  + Handles event of an invalid member ID is provided.
  + Displays member photo, name, role, bio, and a contact email link.

**About.jsx**

* **Purpose:**  
  Presents an overview of SoundScribe and introduces the team.
* **How It Works:**
  + **Company Overview:**  
    Displays a brief description of the service.
  + **Team Grid:**
    - Uses a responsive grid layout to show team member cards.
    - Each card links to the detailed profile view in **CrewDetail**.

## 3. Global Files and Utilities

**App.jsx**

* **Purpose:**  
  Root component that sets up routing and global context.
* **How It Works:**
  + **Routing:**  
    Handles routes using React Router:
    - / for Home
    - /about for About
    - /crew/:id for individual team member details
    - /login and /register for users authentication
    - /profile for user’s profile
  + **Language wrapper:**  
    Wraps the entire webapp with the LanguageProvider to make the language context available in every page.
  + **Navigation:**  
    Includes the global Navbar that remains visible on all pages.

**Utility Files**

* **textUtils.js**
  + **escapeRegExp(string)**  
    Handles special characters in a string so it is safe to use in a regular expression.
  + **highlightText(text, searchQuery)**  
    Searches the text for the user’s query (case-insensitive) and wraps matching segments in <mark> tags for highlighting.
* **utils.js**
  + **extractTitle(filename)**  
    Removes file extensions and common suffixes (like “(Official Video)”) to create a clean title.
  + **formatTime(timeInSeconds)**  
    Converts seconds into a formatted MM:SS string.
  + **groupWordsIntoSentences(words)**  
    Groups individual word objects (each containing timestamp data) into sentences based on punctuation, word count, and pauses between words.
* **LanguageContext.jsx**
  + **Purpose:**  
    Provides a React Context for the current language.
  + **How It Works:**
    - **LanguageProvider** – Wraps the app and manages the language state.
    - **useLanguage()** – A custom hook to access or update the language from any component.

## 4.Entry and Styling

* **main.jsx**
  + **Purpose:**  
    The starting point of the app.
  + **How It Works:**  
    Uses React 18’s createRoot API to render the app within <StrictMode> for development.
* **index.css**
  + **Purpose:**  
    Contains the global styles for the application.
  + **Features:**
    - **Tailwind CSS Imports:**  
      Brings in Tailwind’s base, components, and utilities.
    - **Theming:**  
      Defines CSS variables for light/dark themes and applies them to certain elements.
    - **Component Styles:**  
      Custom styles for navigation, sections, inputs, buttons, scrollbars, and interactive elements.
    - **Responsive Grid Layout:**  
      Defines grid areas and responsive behaviour for different purposes (transcribed text, summary, timestamps, search).
    - **Highlight Styling:**  
      Styles for <mark> tags used in search highlighting.

# 6. תיק משתמש

A logo for a recording studio

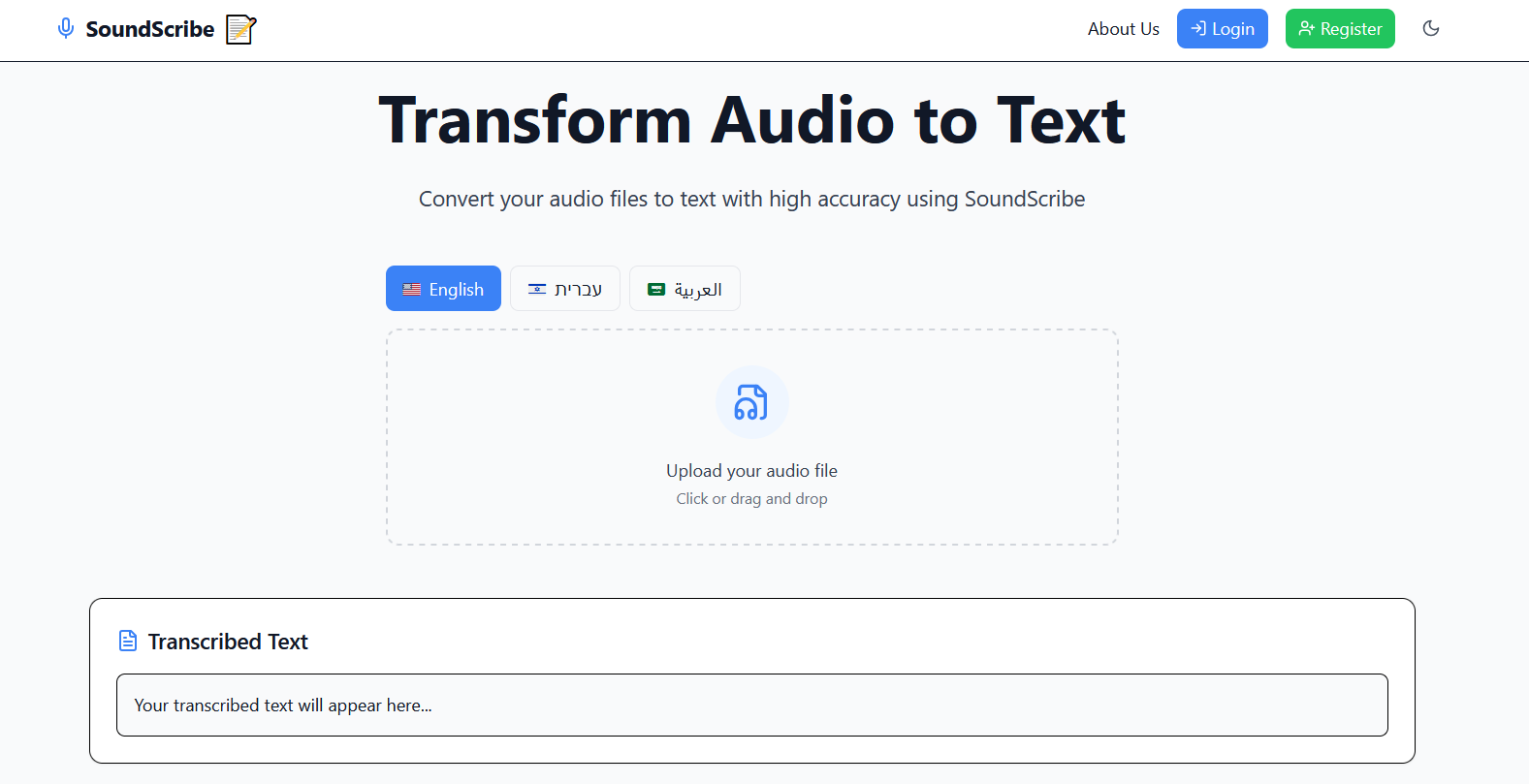
Description automatically generated

# User App Guide

**Quick overview:**

* **Landing page:**  
  Users start at the Home page where they can upload an mp3 audio file.
* **Transcription & Display:**  
  Once the file is processed, the result (including text, summary, words with timestamps) are displayed across several sections on the Home page. Users can use the search bar to search for keywords in the transcription text, the results will appear along with their corresponding timestamps
* **Authentication & Profile Management:**  
  Users can register or log in (via **Register** and **Login** pages). Once logged in, they can view their transcription history in the **Profile** page, where they can select any saved transcription to see details.
* **Feedback & Additional Info:**  
  A star rating system (**StarRating**) collects user feedback. The **About** and **CrewDetail** pages allow users learn more our team.

# 1.Home Page



In the home page, users can choose the language they wish the transcription text and summary to be written at. For best results, it’s best to choose the language corresponding to the one in the audio file uploaded.

A screenshot of a phone

Description automatically generated

To upload an audio file, click on the corresponding area ‘Upload your audio file’. After clicking, users will be asked to choose an appropriate file from their system. Users can also just drag a file and drop it on the highlighted area.

A screenshot of a computer

Description automatically generated

To provide proper feedback to the user, after uploading a file the upload area will change color and display the audio’s raw title. On top of that, the ‘Start Transcription’ button will appear, allowing the user to start the transcription process

A screenshot of a computer

Description automatically generated

To provide further feedback to the user, once the ‘Start Transcription’ button is pressed, it will change color to indicate the file is being processed by the system.

A screen shot of a message

Description automatically generated

After processing, the ‘Transcribed Text’, ‘Summary’ and ‘Sentence Timestamps’ components will be updated to display the audio file’s transcription, summarization and appropriate sentence timestamps.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Users can use the Search Text area to insert a word or sentence they’d like to find in the transcription. The query will then be highlighted in the transcription, summary and timestamp areas. Additionally, in the ‘Search Text’ area will be displayed the query matches from the text, along with their **accurate** timestamps.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedFinally, the home page allows users to rate their experience using our web-app. The user’s vote is saved to the database to be reviewed if necessary.

A white background with black text

Description automatically generatedAfter rating, a message will appear thanking the user.

Afterwards, while the session is still active, the user will not be allowed to vote again.

A white rectangular sign with black text

Description automatically generated

# 2.Login Page

**A screenshot of a login form

AI-generated content may be incorrect.**

Users can navigate to the Login page by clicking “Login” button in the Home page. After the user submits their credentials the system will verify it against the database and if the details are correct, it will display an appropriate message and navigate the user back to the Main page. Otherwise, it will display an error message.

Example of error message for incorrect email address:

A screenshot of a computer

AI-generated content may be incorrect.

Example of error message for missing password:

A screenshot of a computer screen

AI-generated content may be incorrect.

Example of error message for incorrect field:

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

Successful login message:

A white background with black text

Description automatically generated

# 3. Registration page

A screenshot of a login form

AI-generated content may be incorrect.

Users can navigate to the Registration page by clicking “Register” button in the Home page. After the user submits their credentials, the system will validate the data in the fields and if it’s submitted correctly it’ll create a new entry in the database with the user’s information. Afterwards a message will notify the user that the registration was successful, the user will be automatically logged in and navigated back to the Home page. I the details are incorrect, it will display an appropriate error message.

Valid email must have ‘@’ sign and doesn’t previously exist in the database.

Valid password is one with at least 6 letters of any kind.

Successful registration message:

A screen shot of a computer

AI-generated content may be incorrect.

Invalid email address:

A screenshot of a computer

AI-generated content may be incorrect.

Email address already exists:

A screenshot of a computer screen

AI-generated content may be incorrect.

Invalid password:

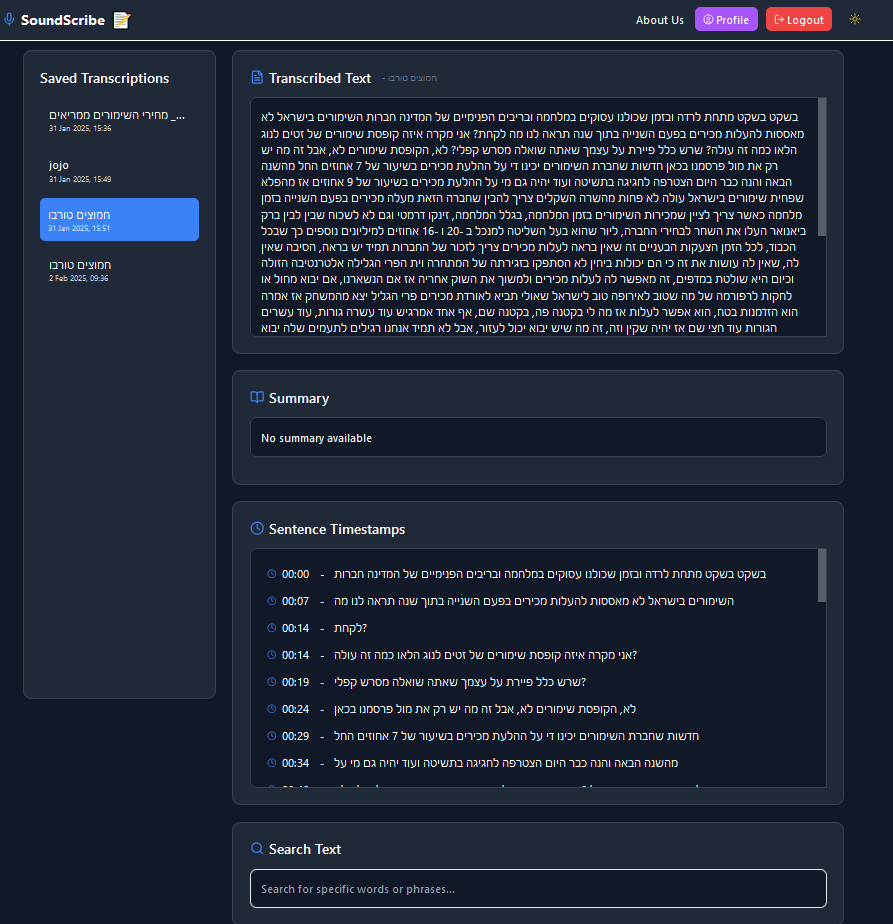
תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

# 4.User profile

After the user signs in through the Login page, the Navigation bar will update to display the ‘Profile’ page. Once accessed, the page displays the user’s transcription history, including the transcription’s text, summary and timestamp and the search functionality present in the Main page. By choosing different titles from the ‘Saved Transcription’ bar, users can go through their transcription history in ease.

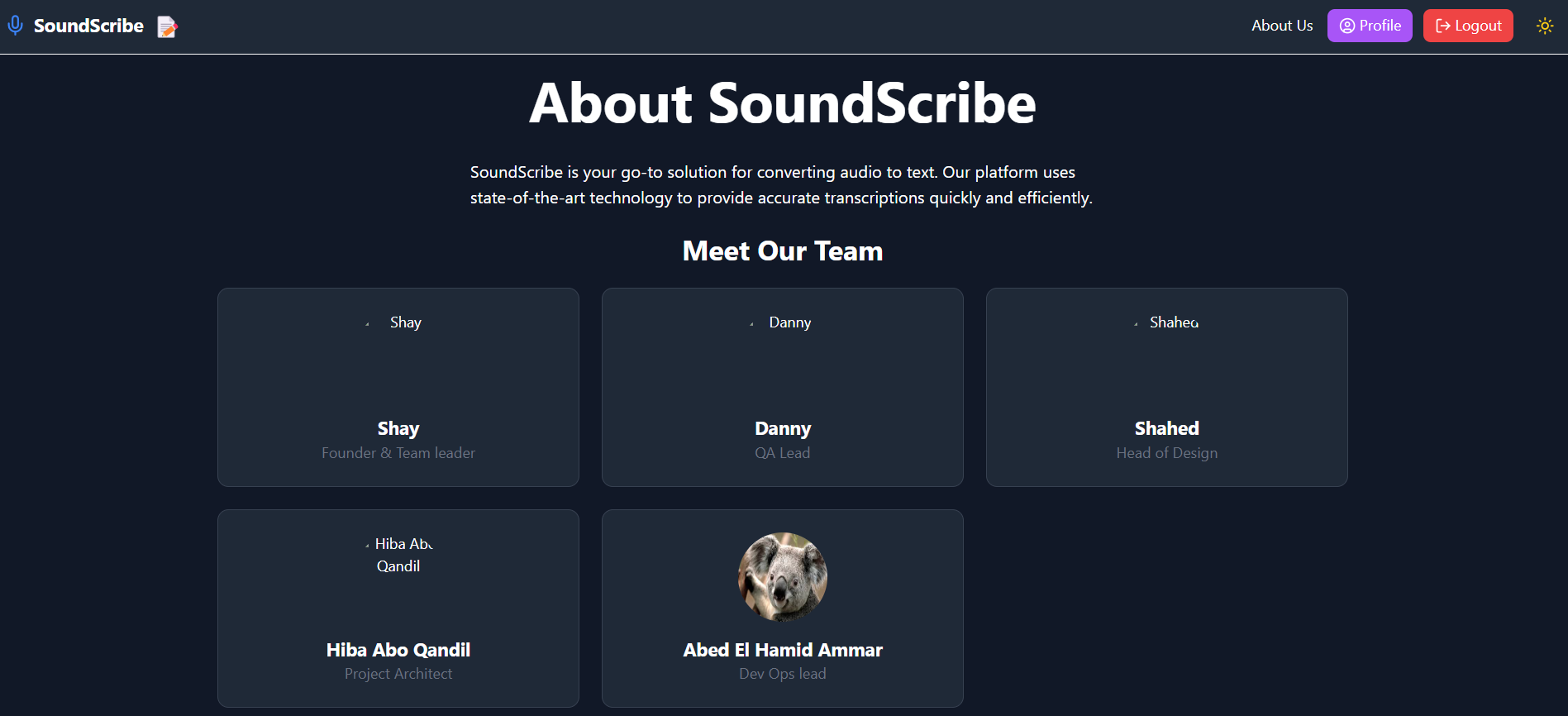
(The following screenshot was taken with the dark theme toggled on to show its effect on the page’s styling)



# 5. About Us page

After clicking the ‘About us’ button from the navigation bar, a page displaying the project’s team and their information will be appear. The page contains clickable profiles of each time member for additional information on them, as well as a short description of the project.

(Again using Dark mode to display styling choices)



Clicking on a team member’s profile will display their personal page with the additional information about them

A screenshot of a computer

Description automatically generated