Project Proposal - AudioScribe

1. Overview of Project:

Project Title: AudioScribe
Project Objectives: The goal of *AudioScribe* is to develop an innovative audio book app that enhances the reading experience. The app's main objective is to provide the users with a convenient way to listen to books by using AI voice technology. The primary objective is to create a platform where users can import books in various file formats, including PDF, EPUB and others, and have the app read the text using an AI-generated voice. The app will accept uploads of all book file types and use an OCR to make all the text readable. From there an AI voice recording will be made reading all of the text and stored in an audio file format for playback. When the recording is finished, a notification will be sent out to let the user know their book is ready to be listened to.

Target Audience: The target audience for the app *AudioScribe* is anyone who enjoys reading or listening to audiobooks.

2. List of Group Members and Their Responsibilities:

Group Members:
[Deval Panchal - 100744653] - Design and Frontend
[Daniel Earley - 100744960] - OCR + File I/O + Text to Speech
[Samuel Bazinet - 100743574] - API and Services

All team members will probably work on all parts of the project, but those are the main responsibilities of each team member.

3. List of Features and Functional Requirements:

File Format: Allow users to import books in various file formats, ensuring versatility and compatibility (PDF, Epub, txt)

OCR: Our app will utilise OCR technology to extract text from documents in cases where the text may be difficult to read due to scanning or other factors.

AI Text to Speech: Use an AI text to speech to generate high-quality, natural-sounding voices for reading the books.

Text Summary: Allow users to get a summary of the text that has been read so far.

Offline Mode: Implement ability for users to download audio versions of books for offline listening.

Accessibility: Include text-to-speech for book descriptions or summaries.

User Accounts: Users can create an account.

Book Information: Users will easily be able to learn information about the book’s author, date of publication, etc.

Book Blurb: Users will be able to read the back cover (aka blurb) of the book they’re about to listen to.

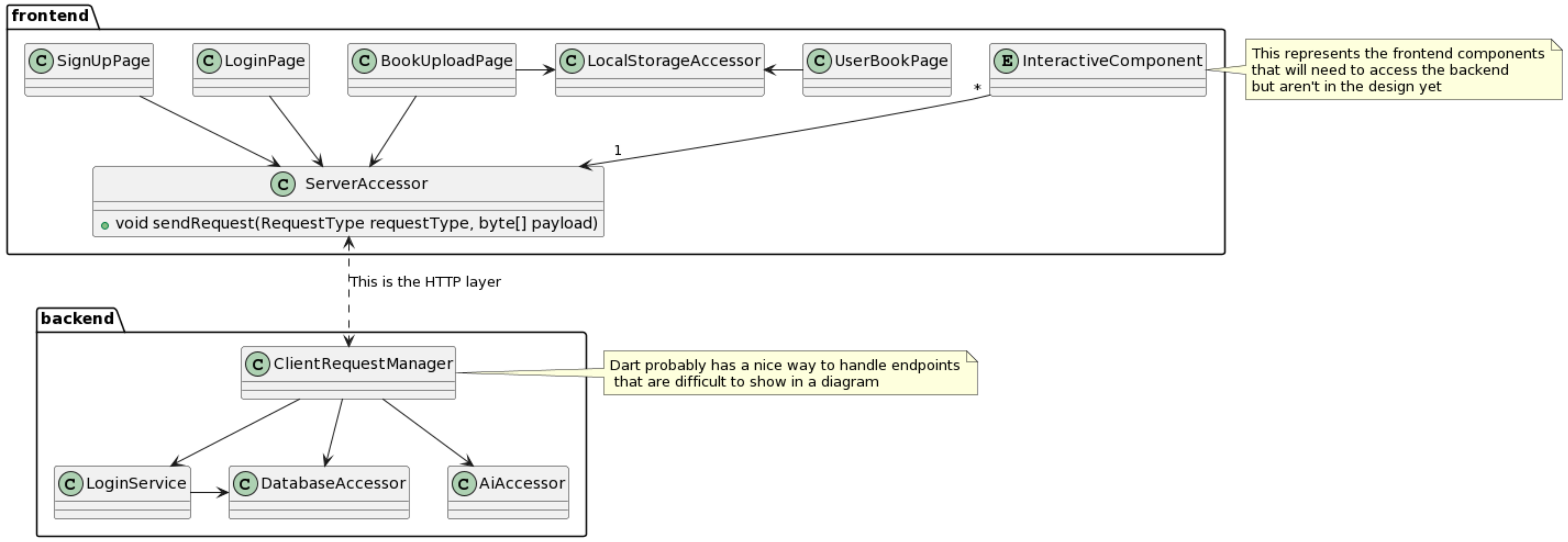
Recommendations: A horizontal recommendations bar will give the users quick access to books they may enjoy.

Library/Saved Books: Users can save books that they have uploaded to the app.

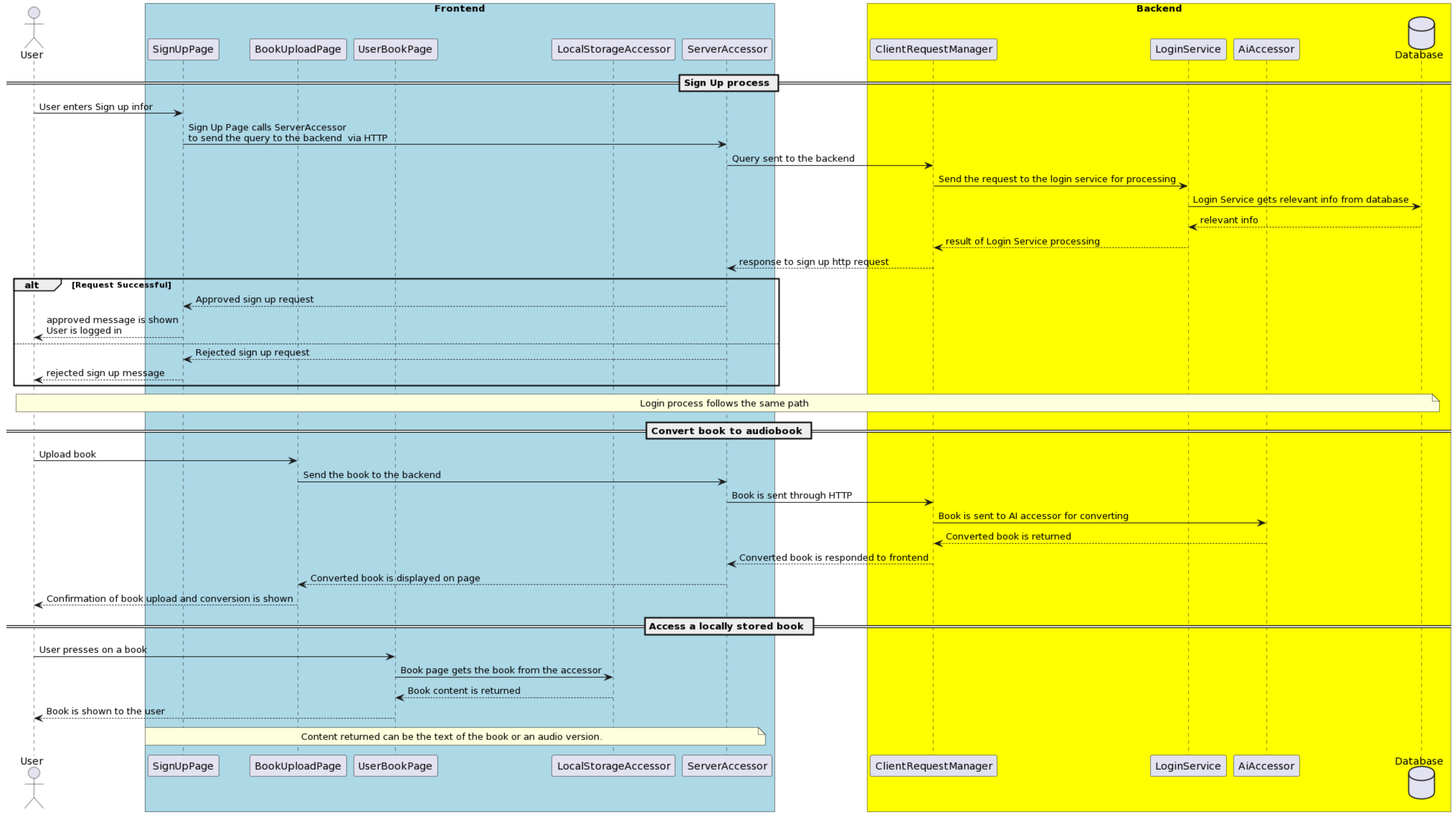
Favourites: Users can favourite a book for quick access the next time they open the app.

4. Code Design (UML or Equivalent) (2.00 marks):

Class Diagram:


frontend
SignUpPage
LoginPage
BookUploadPage
LocalStorageAccessor
UserBookPage
InteractiveComponent
This represents the frontend components that will need to access the backend but aren't in the design yet
*
1
ServerAccessor
void sendRequest(RequestType requestType, byte[] payload)
This is the HTTP layer
backend
ClientRequestManager
Dart probably has a nice way to handle endpoints that are difficult to show in a diagram
LoginService
DatabaseAccessor
AiAccessor

Sequence Diagram:
Common operations handled by tech stack


Frontend
Backend
User
SignUpPage
BookUploadPage
UserBookPage
LocalStorageAccessor
ServerAccessor
ClientRequestManager
LoginService
AiAccessor
Database
Sign Up process
User enters Sign up infor
Sign Up Page calls ServerAccessor
to send the query to the backend via HTTP
Query sent to the backend
Send the request to the login service for processing
Login Service gets relevant info from database
relevant info
result of Login Service processing
response to sign up http request
alt [Request Successful]
Approved sign up request
approved message is shown
User is logged in
Rejected sign up request
rejected sign up message
Login process follows the same path
Convert book to audiobook
Upload book
Send the book to the backend
Book is sent through HTTP
Book is sent to AI accessor for converting
Converted book is returned
Converted book is responded to frontend
Converted book is displayed on page
Confirmation of book upload and conversion is shown
Access a locally stored book
User presses on a book
Book page gets the book from the accessor
Book content is returned
Book is shown to the user
Content returned can be the text of the book or an audio version.

5. Mockup of User Interface (2.00 marks):

Screens:

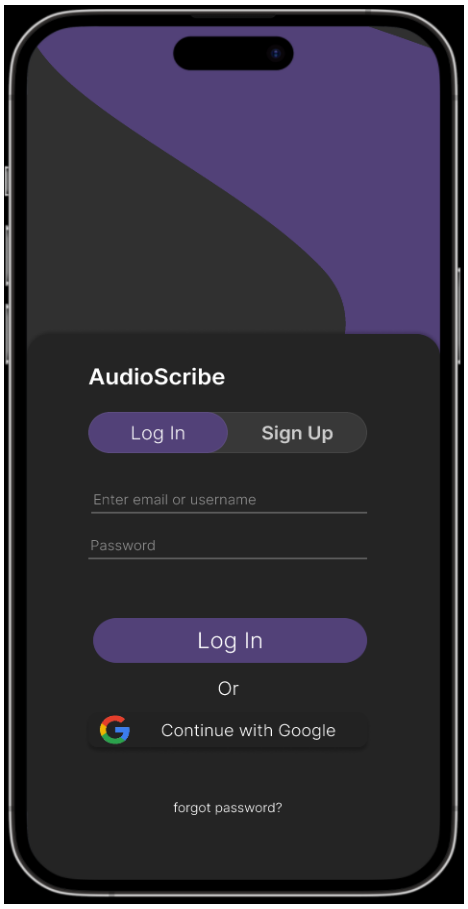


Figure 1

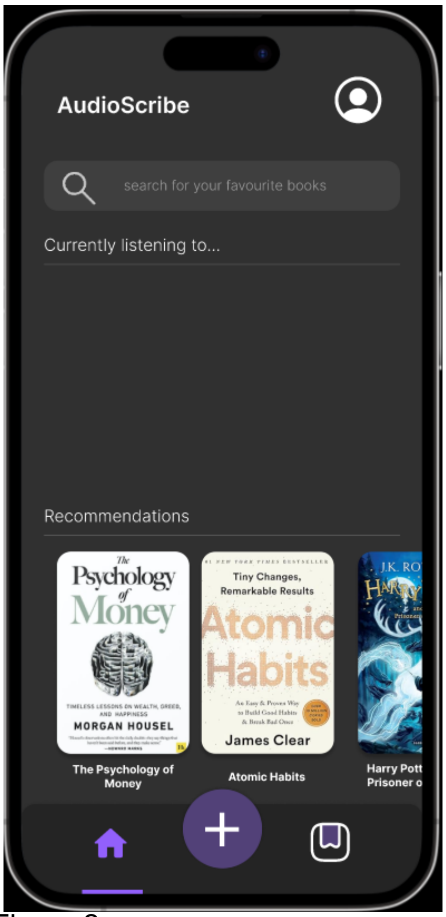


Figure 2

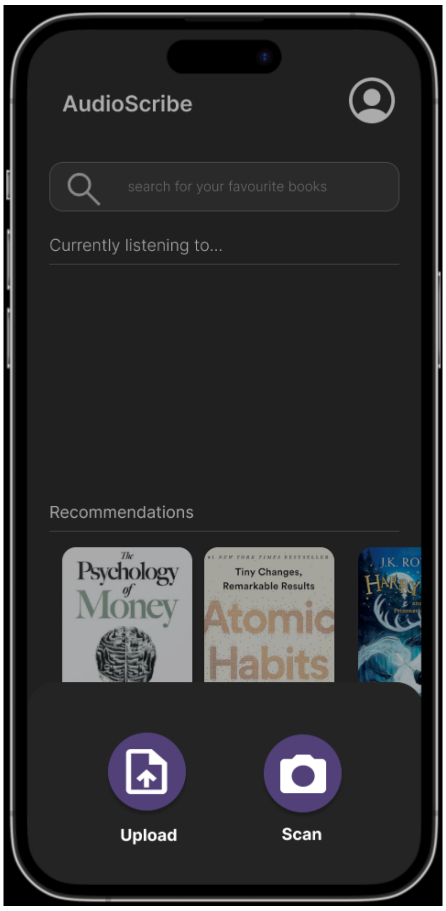


Figure 3

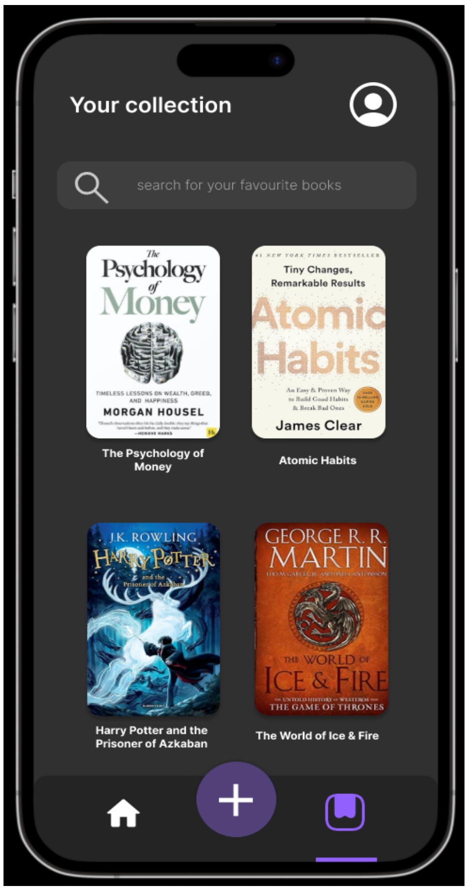
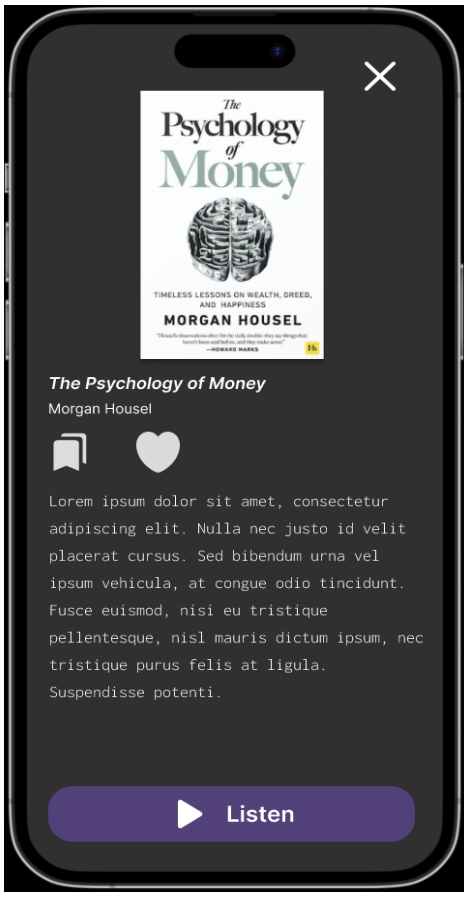
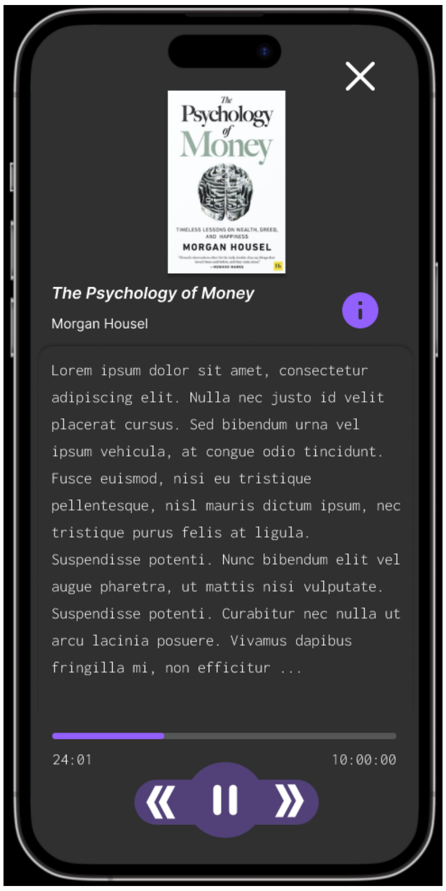


Figure 4



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec justo id velit placerat cursus. Sed bibendum urna vel ipsum vehicula, at congue odio tincidunt. Fusce euismod, nisi eu tristique pellentesque, nisl mauris dictum ipsum, nec tristique purus felis at ligula. Suspendisse potenti.

Figure 5



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec justo id velit placerat cursus. Sed bibendum urna vel ipsum vehicula, at congue odio tincidunt. Fusce euismod, nisi eu tristique pellentesque, nisl mauris dictum ipsum, nec tristique purus felis at ligula. Suspendisse potenti. Nunc bibendum elit vel augue pharetra, ut mattis nisi vulputate. Suspendisse potenti. Curabitur nec nulla ut arcu lacinia posuere. Vivamus dapibus fringilla mi, non efficitur ...

Figure 6

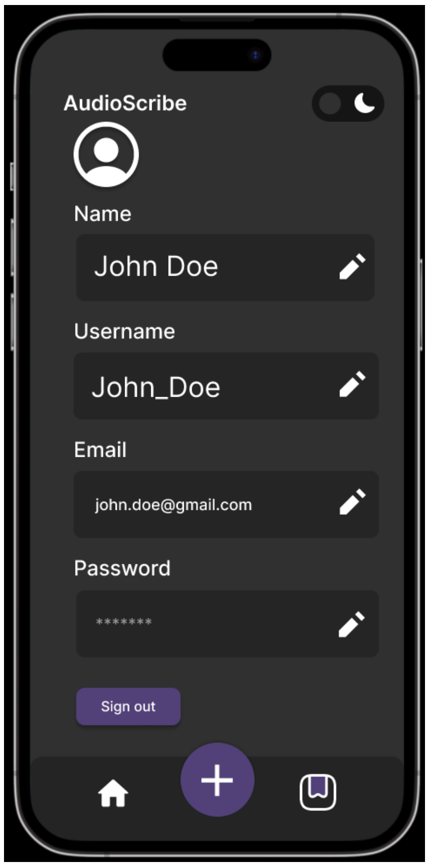


Figure 7

Summary:
To give a quick summary of the mockup, AudioScribe is an app that converts books to audiobooks. Only the major pages have been mocked up, as this encapsulates the overall design and flow for the app, while giving an idea on what additional pages may look like.

- Figure 1:
    - Displays a simple login page, offering multiple modes of sign up, including “Signing in with google”
    - The user can either login or signup from this page by selecting the appropriate button within the selection capsule
    - This page handles the **User Account** portion of the list of features/functional requirements
- Figure 2:
    - Displays the home page, the inspiration for this style of design comes from the “Netflix” design, where user specific and app specific selections are viewed in a horizontal format
    - Creating a simple component for the horizontal navigation may allow a scrollable front page with additional selections, for example genre (Action, Adventure etc…)
- Figure 3:
    - Shows a component that appears when the ‘+’ button is clicked on the bottom
    - Gives user the option to upload documents from their phone or take a picture of a document and have that be read by the app
    - This page handles the **File Format** portion of the list of features/functional requirements
- Figure 4:
    - Shows a page for the saved collection of the user
    - A simple grid layout makes easy selection, along with a search bar that will allow user to navigate a larger collection they have
- Figure 5:
    - Shows page when a book is first selected
    - Showing the title, an option to save/bookmark and like, the summary, and a button which will allow the user to listen to the book contents

- Figure 6:
    - Displays the page when the user is listening to a book or document
    - Allows the option pause/play, forward/backward
    - The info button will access simple metadata on the book → author, publisher, chapters etc…, also allowing the capability to summarise chapters
- Figure 7:
    - Displays the page when the user taps on the profile icon in Figure 2 and Figure 4
    - Allows the user to change their name, username, email and password
    - A toggle button that allows the user to switch between light and dark themes

6. Technology Stack:

Frontend: Flutter framework

- Design Mockup → Figma

Backend: Probably Dart

- AI Voice → Google TTS (basic) → Eleven Labs AI Voice https://elevenlabs.io/, Google Cloud AI TTS, Open source https://github.com/coqui-ai/TTS
- PDF reading → https://products.aspose.cloud/words/dart/conversion/pdf-to-txt/
- Access to OpenAI → https://pub.dev/packages/dart_openai | https://openai.com/product
- OCR → https://medium.flutterdevs.com/ocr-in-flutter-5144ed361239
- File storage of books

Database: SQlite, Firebase

- Can be used to store user account information
- Potential User schema:
    - Username
    - Salted Encrypted password
    - Date of account creation
- Potential Book schema:
    - Title
    - Author
    - Date of publish
    - Location on server storage (C:\\local\...)