TED UNIVERSITY

CMPE 492 / SENG 492 Senior Project

<<BooTunes>>

Final Project Report

Spring 2025

Team Members:

Mehlika Eroğlu, 58717180232, Software Engineering,

Melih Aydın, 28180576276, Computer Engineering,

Batuhan Dalkılıç, 11491297756, Computer Engineering,

Elif Alptekin, 31376233198, Computer Engineering

Supervisor: Venera Adanova

Jury Members:

Gökçe Nur Yılmaz

Eren Ulu

Contents

1. Executive Summary

BooTunes is an innovative application designed to emotionally enhance the digital reading experience. By analyzing each page of a PDF-format book uploaded by the user using Natural Language Processing (NLP) techniques, the system identifies the dominant emotion and plays a corresponding music genre. This personalized synchronization between emotion and music aims to create an immersive reading environment. The project combines real-time sentiment analysis, a user-friendly interface, and modular backend components to deliver a scalable, flexible, and intelligent multimedia experience.

2. Introduction

2.1. Project Background & Motivation

While digital reading has become increasingly popular, personalization in emotional engagement has been limited. BooTunes addresses this gap by transforming reading into a multi-sensory experience. Leveraging advancements in NLP and AI, BooTunes dynamically detects the emotional tone of each page and plays music that aligns with it. This approach not only strengthens emotional connection to the content but also enhances user immersion and satisfaction.

2.2. Problem Definiton

Current digital reading platforms do not offer emotional synchronization, resulting in a static experience. There is no existing system that adjusts background music based on the mood of the text being read. BooTunes solves this by performing real-time emotion analysis per page and dynamically changing the music to reflect the detected sentiment, such as joy, sadness, or suspense, thereby enriching the emotional experience of the user.

2.3. Target Audience

The primary audience includes frequent digital readers who seek a more immersive and personalized reading experience. Literature enthusiasts, students, and emotionally-aware readers who value dynamic engagement are key users. Additionally, people interested in combining music and reading for relaxation or educational purposes may also benefit from the application.

2.4. Scope and Limitations

Scope:

- The system supports PDF file input and performs page-by-page sentiment analysis.
- A suitable music genre is selected and played according to the detected emotion.
- A web interface (React.js) and partially developed mobile version (React Native) are included.
- Users can control music playback and customize settings like preferred genres or themes.

Limitations:

- Currently limited to English-language content.
- Sentiment detection accuracy depends on the performance and limitations of the NLP models used.

- Music is selected from a predefined dataset; integration with streaming APIs (e.g., Spotify) is under development.

- The summarization feature focuses on factual or surface-level content extraction and lacks the nuance and literary interpretation expected from advanced human-level summarization. Therefore, summaries may not reflect deep narrative tone, context, or literary subtleties.

- Mobile app development is in progress and may not include all features from the web version.

3. Final System Architecture and Design

3.1. System Architecture

BooTunes is structured using a **Model-View-Controller (MVC)** architecture with a focus on modularity and scalability:

- **Model Layer** handles user data, book metadata, emotional analysis, and music/visual management.
- **Controller Layer** processes API requests, executes logic, and communicates with models.
- **Client (View) Layer** provides the user interface using React/React Native.

Key Components:

- **EmotionAnalyzer**: Applies NLP models to detect emotions in book text.
- **MusicManager**: Matches detected emotions with suitable music tracks.
- **VisualManager**: Displays mood-based visual effects.
- **RecommendationEngine**: Offers personalized content based on user behavior and preferences.

3.2. Design Details

NLP Strategy: A lightweight NLP model is used for real-time page analysis, while a deeper model runs asynchronously for refinement.
Modularity: Components such as EmotionAnalyzer and MusicRecommender are independently designed and loosely coupled.
Database: MongoDB is used to handle user preferences and reading history with flexibility.
Interface: A clean and responsive UI supports seamless transitions and real-time updates.
User Customization: Users can set preferences for genre, theme, brightness, and more through the Settings module.

4. Engineering Impact Analysis

The BooTunes project aimed to explore how current technologies such as NLP, music analysis, and multimedia synchronization could be integrated into a digital reading environment to create a more engaging experience. While the system is still at a prototype stage, many of the engineering decisions made throughout the development process were guided by scalability, accessibility, efficiency, and user-centered design. In this section, we evaluate the potential impact of these design choices across global, economic, environmental and societal dimensions.

4.1. Global Impact

Although BooTunes is currently designed and tested in English, the system architecture supports potential adaptation for multilingual use. We used cross-platform tools like React, React Native, and Firebase to ensure that the application could eventually be accessed by users from various regions and device types. Additionally, the integration of zero-shot NLP models suggests the possibility of scaling the emotion analysis component without needing extensive labeled data and retraining the large models in different languages. While we were not able to implement multilingual support during this phase, the architecture could accommodate it in future work.

4.2. Economic Impact

Our system demonstrates a scalable, low-cost model for integrating AI and entertainment technologies in digital publishing. As a student team working within limited time and resources, we prioritized tools and design choices that were accessible, affordable, and practical for rapid prototyping. Our goal was to demonstrate that it is possible to experiment with emotionally intelligent systems without relying on expensive or enterprise-level infrastructure.

To that end, we used open-source technologies such as React, Firebase, and PyTorch. These tools allowed us to build a functioning system without incurring significant costs, while also gaining hands-on experience with widely used industry technologies. For example:

- Firebase handled authentication and real-time data storage efficiently during development.

- Hugging Face offered free access to powerful NLP models, which we used for zero-shot emotion classification.

Although we initially planned to use the Spotify API, limitations in access and usage constraints led us to implement a custom local music database. This provided a controlled environment for testing and allowed us to tag music with emotion categories ourselves. In a real-world scenario, this approach could also reduce dependency on external APIs and potentially benefit smaller developers or publishers.

4.3. Environmental Impact

Even though BooTunes is a small-scale project, we considered sustainability in our technical design. The application promotes digital reading over printed materials, which aligns with the general environmental goals such as reducing printing and packaging waste. To reduce unnecessary API usage and improve performance, we implemented client-side caching and kept inference processes as lightweight as possible.These steps may have a modest impact in our prototype but reflect our awareness of energy efficiency and scalable software practices.

4.4. Social Impact

The project aims to offer an engaging and emotionally responsive reading experience. By aligning music and visuals with the emotional tone of texts, the system could potentially enhance user immersion and encourage reflection on narrative mood. Although our current implementation has limited depth in terms of literary nuance, early feedback suggests the concept is promising.

We were particularly interested in how emotional cues could be used to enrich a reader's experience. While our emotion recognition is not perfect, and results vary depending on the text type, we were able to create a system that links music and visuals to the general mood of a passage.

This has potential future applications in:

- Education which help students analyze emotion tone in texts
- Mental wellness, by enhancing emotional awareness and engagement
- Assistive technologies, especially for readers who benefit from multimodal feedback

In addition, the project opened discussions about how such tools could be adapted for educational or accessibility-focused contexts. For example, readers with learning differences might benefit from synchronized audio-visual cues. While these ideas remain speculative, they point to directions in which the project could evolve.

We were also mindful of ethical and privacy considerations, particularly given the growing use of affective computing. Our system processes emotion analysis locally without storing personal emotional data. Although we did not implement comprehensive privacy protocols, our architecture allows room for responsible handling of user information in future versions.

5. Contemporary Issues Discussion

5.1. Ethical Use of Emotion-Based AI

One of the core features of BooTunes is emotion analysis both from text and from music. Although this adds a personalized and engaging layer to the user experience, it also raises questions about the ethical use of emotional data.
We recognize that affective computing can influence user mood and behavior, and this influence must be applied carefully. In our case, we do not track or profile users emotionally; rather, the emotion detection applies only to the content being read, not the reader themselves. However, if future versions were to include personalized mood tracking, we would need to implement clear consent mechanisms and consider the psychological impact of emotion-based suggestions.

5.2. Privacy Considerations

Our project uses Firebase Authentication for login and a Realtime Database to store personalized music preferences or playlists. While this system was sufficient for our prototype, it raised questions about how user data should be handled in more mature applications.

Currently, BooTunes does not collect sensitive information or perform analytics on user behavior. However, we acknowledge that secure storage, transparency, and user control over data would be essential in any production-grade system. Privacy risks could increase if more detailed usage tracking or emotional feedback were implemented, so future iterations must apply privacy-by-design principles and possibly include local-only processing options.

5.3. Challenges with Music Licensing and API Limitations

Initially, we intended to use the Spotify Web API to deliver music aligned with the emotional tone of the text. However, due to API limitations and content access restrictions, we were unable to integrate full playlist streaming as planned. Instead, we created a local music repository to ensure functionality during testing.

This situation highlighted a broader contemporary issue: developers are often constrained by licensing policies, quota limits, or closed platforms when trying to innovate. It also raised awareness about the challenges of developing content-driven applications that depend on third-party platforms. If BooTunes were to be released publicly, further attention would be needed to comply with licensing terms, and we might need to explore open audio platforms or seek partnerships.

5.4. Model Interpretability and Bias in Emotion Analysis

We used models to analyze emotional tone in text. While these models are flexible and do not require labeled training data, they are also difficult to interpret and occasionally produce ambiguous or overly simplistic classifications.

This limitation reflects a broader issue in AI today: the trade-off between accessibility and explainability. Literary texts often contain layered emotions, sarcasm, or cultural nuance that general-purpose models may miss. Although our prototype operates at a basic level, we understand that in more advanced applications, model bias, overgeneralization, or lack of transparency could undermine user trust or misrepresent content.

6. New Tools and Technologies

In this project process, current and powerful technologies were used to increase the user experience and to ensure the flexibility and sustainability of the system. Below, the basic tools and technologies used in the development process are explained in detail along with their contributions to the project.

- **React + JSX + Tailwind CSS:**

We used the React library to create our user interface. React’s component-based structure allowed us to develop our application modularly. We were able to write HTML-like structures with JavaScript using the JSX syntax in React. This structure increased both readability and accelerated the development process. We used Tailwind CSS in the interface design. Thanks to this utility-first CSS framework, we performed styling operations directly on HTML-like structures and did not have to write complex CSS files.

- **Firebase Authentication and Realtime Database (continued):**

We will use Google Firebase technologies to allow users to log in securely to the system and store their personal playlists. With the Firebase Authentication module, users can register or log in with their email and password. We aim to use the Firebase Realtime Database to store data. In this way, user-specific music preferences or playlists they create can be saved and recalled in real time.

- **Music Database (Local Use):**

Although we initially planned to use the Spotify Web API, we are now more likely to create our own music repository to test our mood recommendation system with a more controllable dataset. In our tests, we observed that we can connect to the Spotify API but cannot access the playlists. Therefore, we designed a music repository that includes the names, categories, and access links of the music so that we can guarantee backup. Although we have guaranteed the integration of our music repository with the project, we will do more tests for the Spotify API. It can be expanded with external APIs in the future if necessary.

- **Stable diffusion (Image Creating):**

For the image generation subsystem, we used the most efficient stable diffusion model, and although our test results were not exactly what we wanted, we use this model because the best images we obtained belonged to this model.

- **Rag, Pegasus, Bart, Deepseek (Summary Creating):**

We used many models for the summarization subsystem and were able to obtain a system that could summarize. However, since the summary is not literary in literary texts such as poetry, we could not get our summaries exactly as we wanted. For this reason, we are trying models such as Rag, Pegasus, Bart and Deepseek, and we will continue with the model that gives the best result.

- **CUDA + PyTorch:**

NVIDIA CUDA-supported hardware was used to run deep learning models more efficiently. Both text and audio-based emotion recognition models were developed with the PyTorch framework. Thanks to CUDA, training and inference processes were accelerated, which positively affected real-time performance.

- **Zero-Shot Learning + facebook/bart-large-mnli(Emotion Analysis):**

The zero-shot classification method was adopted in the text analysis part of our project. For this purpose, the facebook/bart-large-mnli model was used. With this model, sentiment prediction could be made on unlabeled texts and categories
suitable for the user could be determined.

- **Essentia + Custom Audio-Emotion Mapping:**

The Essentia audio processing library was used for audio-based emotion analysis. With this library, a special emotion-music mapping algorithm was developed by analyzing the acoustic properties of the music (tempo, energy, danceability, etc.). Thanks to this custom mapping system, the emotion contained in the music was matched more harmoniously with the user's detected mood.

7. Research and Resource Utilization

In order for our project to progress successfully and with a solid foundation, we conducted a comprehensive literature review and source research. Throughout the development process, we shaped our technical decisions by using both academic and sectoral sources and created the theoretical and practical knowledge infrastructure we needed. This section details the main research methods we used, the types of sources, and their impact on the project.

7.1. Use of Academic and Theoretical Resources

In order to develop the project based on scientific foundations, especially in the areas of sentiment analysis, recommendation systems and user experience, prestigious publications such as IEEE and ACM were used. Through these sources, text and audio-based sentiment analysis methods, deep learning-based classification algorithms and current research on the music-emotion relationship were examined. The following topics were particularly focused on:

- Comparison of transformer-based models for sentiment analysis
- Use of zero-shot learning methods in recommendation systems
- Psychological effects of music genres and sentiment mapping
- Recommender System architectures and user profiling
- User-friendly interface design principles (UI/UX) in web-based applications

7.2. Applied Knowledge Resources

During the project, we frequently used official documentation, open source developer forums, GitHub sample projects and platforms such as Stack Overflow to access information about the application. These resources were very important for us, especially in the technical integration details of React, Firebase, PyTorch and Microsoft Azure services. In addition, the official HuggingFace documentation of the models used (such as BART-MNLI, Emotion-RoBERTa) was carefully examined, and in-depth research was carried out on topics such as model input/output formats and preprocessing requirements and joint action was taken.

7.3. Library and Dataset Usage

We directly examined the ready-made models and libraries used in the project (e.g. facebook/bart-large-mnli, j-hartmann/emotion-english-roberta-large, essentia) on HuggingFace, PyTorch Hub and GitHub and learned what we needed to do and how to do it and continued. We also searched for datasets that would allow matching music to emotion categories; first we evaluated resources such as "DEAM: Database for Emotional Analysis in Music" and "Million Song Dataset"; however, we thought that a more specialized dataset would be more efficient, so we chose songs from these datasets and our own playlists to create our own tagged music pool.

7.4. External Sources

In this process, we made significant progress thanks to the people we got help from and consulted, in addition to written sources. First of all, we talked to our project supervisor, Venera, at regular intervals about our progress, the parts we had difficulty with, and the parts we could improve, and we made faster progress by getting feedback from her. Also, since many people in our group were doing internships, we had the chance to ask questions to our superiors at the internship sites and solve small problems when we got stuck.

This entire research and resource usage process ensured that the project was not only functional but also scientifically based. We believe that this system, which we created by blending academic and practical knowledge, is a valuable prototype for the future of AI-supported music recommendation platforms. Thanks to the follow-up of relevant literature and the correct use of technical resources, we were able to produce effective and fast solutions to the problems we encountered and left the project open to easy development in the future.

8. Testing and Results

8.1. Test Plan Overview

The testing strategy of the BooTunes project has been designed to ensure that the system is robust, functionally accurate, and capable of delivering seamless user experience across various usage scenarios. BooTunes aims to enhance digital reading by analyzing the emotional tone of texts and matching it with appropriate music and visuals. Given the interdisciplinary nature of this project—combining NLP, emotion classification, music synchronization, and user interaction—testing covers both technical correctness and experiential quality.

The test plan encompasses multiple levels of testing, including unit testing, integration testing, system testing, performance testing, user acceptance testing (UAT), and beta testing. Each phase focuses on different validation aspects:

- Unit tests ensure that individual modules like emotion detection, music selection, visual generation, and PDF parsing function correctly in isolation.
- Integration testing validates the interaction between modules (e.g., EmotionAnalyzer → MusicManager), as well as communication with external services such as the Spotify API.
- System testing examines the entire application flow in realistic user scenarios, validating end-to-end behavior such as uploading a PDF, emotion analysis per page, audio-visual synchronization, and personalization.
- Performance testing evaluates how the system handles stress conditions like large files, slow networks, or offline mode, focusing on response times, resource usage, and fallback behavior.
- User Acceptance Testing (UAT) assesses whether the application meets real user expectations and delivers a meaningful emotional reading experience.
- Beta testing extends real-world validation to a broader user group, identifying unexpected usage patterns, device-specific bugs, and subjective feedback on

emotional impact.(Since we are in developing stage we haven't applied the UAT and beta testing yet)

The testing methodology leverages both manual and automated tools. Frameworks such as pytest and Jest are used for unit and integration tests, while tools like Postman, React Testing Library, and browser/mobile debuggers support API and UI validation. Mocking strategies are applied for Spotify API and emotion detection to simulate conditions like network failure or data unavailability. The test environment includes both desktop-based setups and planned mobile testing using React Native.

Each test case is carefully structured with a unique ID, input conditions, expected output, and status, ensuring traceability and regression tracking. Control procedures are implemented through GitHub Issues to document, assign, and resolve bugs systematically.

Through this layered and iterative test plan, BooTunes aims to validate the core functionality, improve system reliability, and ensure that users can enjoy a fluid, emotionally adaptive multimedia reading experience on both web and mobile platforms.

8.2. Test Cases and Results

1. Unit Testing

Unit testing focuses on verifying the correctness of individual functions, methods, or classes in isolation from the rest of the application. In the BooTunes system, these units belong to subsystems such as emotion analysis, music selection, visual generation, user interaction, and data management.

The objective is to ensure that each building block performs its intended logic under various conditions, including edge cases, erroneous input, and boundary values.

Scope of Unit Testing

Subsystem	Module / Class	Unit Test Target
Emotion Analysis	EmotionAnalyzer	Detecting correct emotion label from text input Handling ambiguous or multi-emotion sentences

Emotion Analysis	TextPreprocessor	Cleaning and tokenizing raw text Removing noise (punctuation, HTML tags, etc.)
Emotion Analysis	EmotionLabelMapper	Mapping model output scores to standardized emotion categories
Music Management	MusicSelector	Mapping emotions to genre correctly Selecting the correct track from playlist pool
Music Management	PlaybackController	Play, pause, skip logic Handling edge cases (empty playlist, repeated songs)
Music Management	VolumeManager	Adjusting volume levels Toggling mute/unmute states
Visual Management	VisualGenerator	Generating appropriate visuals for given emotions Caching and fallback behavior
Visual Management	ThemeSelector	Assigning visual themes to each emotion
PDF Processing	PDFParser	Extracting clean text per page Handling corrupt or non-standard PDFs
PDF Processing	PageNavigator	Page-forward and page-backward navigation logic
User Interaction	BookmarkManager	Saving and retrieving bookmarks Handling boundary conditions (first/last page)

User Interaction	SummaryGenerator	Generating accurate and concise summaries Handling edge cases (short books, disconnected text)
User Interaction	UserPreferenceHandler	Updating and retrieving user preferences (genre, emotion intensity)
Data Layer	DatabaseHandler	Insert, read, update, delete operations (CRUD) for bookmarks, summaries, and playback data
Data Layer	SessionManager	Managing per-user reading sessions and state
API Integration	SpotifyService	Fetching playlists, refreshing tokens, handling Spotify errors
API Integration	EmotionModelAPI	Sending requests to the NLP model Parsing and validating emotion predictions
Utility Functions	TextCleaner, Logger	Text normalization Log formatting Helper method accuracy

2. Integration Testing

Integration testing is essential for BooTunes due to its modular and API-dependent architecture. It verifies that separate components—such as emotion analysis, music selection, and UI—interact correctly and consistently when combined. This phase follows unit testing and ensures seamless data flow, correct logic execution, and real-time responsiveness.

Given BooTunes' reliance on external services like Spotify and AI models, integration testing helps identify issues such as API failures, latency problems, or interface mismatches that are not visible in isolated testing.

Key goals include:

- Validating module-to-module and API communications
- Ensuring user actions result in accurate backend responses
- Verifying fallback mechanisms under network/API failure
- Confirming that user preferences are correctly applied system-wide

BooTunes uses an interface-driven integration strategy, covering:

- **Internal Integration:** Backend modules (e.g., EmotionAnalyzer → MusicManager)
- **External Integration:** APIs like Spotify and Hugging Face
- **Frontend-Backend Sync:** React-based UI reflecting backend logic in real time

This structured approach ensures overall system stability and cohesive user experience.

Core Integration Points

Integration Point	Integration Description	Expected Outcome
BookController → EmotionAnalyzer	**Parsed book text sent for emotion classification**	**Dominant emotion correctly identified per page**
EmotionAnalyzer → MusicManager	**Emotion label triggers genre-based track selection**	**Audio track selected matches emotion and respects user preferences**
EmotionAnalyzer → VisualManager	**Emotion metadata triggers themed visual rendering**	**Visual theme reflects emotional tone or**

		fallback to user-defined default
UserPreferencesController → All Components	**User-defined settings (e.g., genre, brightness) applied to downstream modules**	**Preferences are respected across modules; rendering and audio behavior aligns**
Backend → Spotify API	**API calls made to retrieve music tracks based on detected emotion and genre**	**API returns expected track metadata, or fallback is used on error**
React/React Native UI → Backend	**UI requests emotion/audio/visual for page navigation or updates**	**Backend processes request and responds with appropriate content for immediate use**

3. System Testing

System testing aims to validate BooTunes as a fully integrated application under real-world conditions. It checks whether all components—emotion analysis, music and visual synchronization, personalization, and bookmarking—work together seamlessly across platforms. This phase ensures that the app meets both functional and user experience expectations, confirming end-to-end consistency and stability during actual usage.

Test ID	**Feature**	**Test Scenario Description**	**Input/Action**	**Expected Result**
ST-001	**PDF Upload**	**Ensure the system parses PDF files correctly**	**Upload 100-page book in .pdf format**	**Text content is parsed and displayed page-by-page**
ST-002	**Emotion Detection**	**Detect emotion across multiple pages**	**Navigate through 5 pages of emotionally varied text**	**Detected emotions vary by page and match sentiment**
ST-003	**Music Matching**	**Change music based on emotion detected**	**Page with 'sad' sentiment**	**Calm/sad genre music is played**
ST-004	**Visual Theme Adaptation**	**Display a background visual based on the overall emotion or genre of the book**	**Open a book categorized under 'nature' or 'melancholy' genre**	**A nature-themed or emotion-appropriate visual is loaded once and remains consistent**
ST-005	**Offline Mode**	**Test playback and visuals without internet**	**Open pre-downloaded book in airplane mode**	**No error, music and visuals work seamlessly offline**
ST-006	**Summary Feature**	**Generate summary of previously read content**	**Tap 'Summarize Last 5 Pages'**	**Accurate 2–3 sentence summary is generated**

4. Performance Testing

Performance testing in BooTunes has focused on key features that are currently implemented and testable, such as emotion analysis, summary generation, and PDF processing. The goal is to evaluate how smoothly and quickly the system responds under typical usage.

In our current testing:

- **Emotion analysis** across an entire book (around 100 pages) initially took nearly **3 minutes**, but after optimizing the preprocessing and batching structure, it was reduced to around **1 minute**.
- **Summary generation** for 5–10 pages of content initially required about **3 minutes**, depending on the density and length of the text. After using a lighter model, average time dropped to around **50 seconds**.
- **PDF uploads** and parsing operations (for books over 100 pages) were completed in under **20 seconds**, with each page rendered correctly and without lag during navigation.
- Basic frontend actions like **page switching** and **updating visual backgrounds** responded within **1–2 seconds** in desktop environments.

These results show that the implemented features of BooTunes currently offer stable and responsive performance for core use cases. As more modules (e.g., Spotify integration, personalization logic) are completed, further performance tests will be conducted.

5. Applied Tests and Results

This section presents the functional test results of the BooTunes application. The listed test cases reflect the actual development status of the system, covering both successful modules and those still under construction. Features like emotion analysis, offline playback, and PDF parsing performed well, while personalization and full backend sync require further development. Each entry outlines the test ID, feature tested, input and expected behavior, actual result observed, and final status.

Test ID	Feature	Test Description	Input	Expected Output	Actual Result	Status
TC-001	Emotion Analysis	Detect dominant emotion from chapter text	Chapter 1 content	Emotion: Sadness	Emotion returned: Sadness	Pass
TC-002	Emotion Analysis	Handle unsupported language or unreadable text	Page with OCR errors	Return neutral or warning	Returned: Neutral	Pass
TC-003	Music Matching	Play correct genre for 'Joy' emotion	Emotion: Joy	Pop or upbeat track is played	Upbeat track started	Pass

TC-004	Music Matching	Fallback to curated music if Spotify fails	No Spotify connection	Play from local emotion-mapped database	Fallback music played successfully	Pass
TC-005	Visual Generation	Display correct theme image based on book setting	Book part of a place	This image of place is displayed	Image of this place is displayed	Pass
TC-006	Visual Generation	Display correct theme image based on book setting	A dialog of two people	Two people which are talking image	Strange faces and environment that are a combination of several images	Fail
TC-007	Summary Feature	Generate summary for selected chapter	Chapter 2 text	Brief summary that captures essence & tone of the chapter	Summary generated but lacks structure	Partial
MT-001	UI Login	If Login is successful or not	Email and password	Succesful login and navigate to the Homepage	Succesful login made and navigated to the Homepage	Pass
MT-002	Settings Retention	Test whether user preferences are saved after reload	Set preferences (e.g., genre: Jazz) → Reload page	Preferences are remembered, jazz music plays	Preferences retained	Fail

8.3. Test Assessment

Testing efforts for BooTunes have been primarily focused on the core functionalities that have reached a near-complete or usable state. Based on current progress, several components of the system have shown reliable results, while others remain under development or require further validation.

Stable and Successfully Tested Areas:

- **User Interface:** The UI is nearly finalized, offering a clean and responsive experience. Navigation, layout structure, and interactive elements (e.g., dark mode, page transitions, bookmark buttons) have been tested and perform consistently across desktop platforms.
- **Emotion Detection:** The emotion analysis module has been successfully integrated and tested using real book content. It accurately identifies emotions and reacts within an acceptable time frame. Performance has improved over multiple iterations, ensuring responsiveness even in longer texts.
- **PDF Parsing and Page Rendering:** Large book uploads (100+ pages) were parsed and rendered correctly page-by-page, with minimal loading delays and no crashes.

Partially Tested / Still Developing:

- **Music Recommendation:** Basic music matching emotions is implemented using a local, pre-classified dataset. While functional, real-time Spotify API integration and user preference-based personalization are still under development and not yet testable in full. Fallback logic has been tested manually but needs broader coverage.
- **Visual Generation:** Static visuals are shown correctly based on emotion or book theme. However, the AI-generated or dynamic visual adaptation system is still in design, and its performance hasn't been evaluated.
- **Summarization:** The current summarization model is usable and generates coherent results for shorter sections. However, it struggles with longer or complex narratives. Further testing is pending in the integration of an improved model.

Not Yet Tested / In Progress:

- **Final Backend Integration:** Full backend coordination—including API request handling, emotion-music sync, personalization propagation, and offline cache logic—is still in progress. As such, true end-to-end synchronization between frontend actions and backend responses cannot yet be validated.

- **Mobile Platform Testing**: While the interface is built to support cross-platform use, testing on mobile devices is scheduled for the next phase. Performance and UI responsiveness on varied screen sizes and hardware must still be evaluated.

In summary, the BooTunes system has shown strong functional behavior in emotion detection, UI interaction, and static content rendering. Modules that depend on backend coordination or dynamic real-time services are still under development, and their testing will be prioritized in upcoming milestones. Current test outcomes provide a solid foundation for focused improvements and integration in the next sprint.

9. Conclusion and Future Work

The BooTunes project, designed to enhance the digital reading experience through real-time emotion-based music and visual integration, has made significant progress in both front-end implementation and backend architecture planning. Currently, the project has reached a semi-functional prototype phase, with substantial advancement in the **user interface** and **emotion analysis modules**, while some critical backend and integration components are still under development.

Project Progress Overview:

As of now, the user interface has been largely developed with React and React Native, achieving a visually rich, responsive, and cross-platform layout. The Home, Reader, Settings, and BookLibrary pages are implemented with functional elements like dark mode, dynamic playlist selections, and page navigation. These components provide the foundational interaction experience intended for the final product.

On the machine learning side, the emotion analysis subsystem is fully operational. A suitable pre-trained NLP model was integrated and tested using classical books like *Anna Karenina* and *Siddhartha* to verify the system’s ability to detect emotional patterns. The system was successful in dynamically identifying emotions and reacting to them with matching outputs in test environments.

However, the music synchronization module is currently under active development. While basic matching between emotion and music genres is implemented using a local pre-classified dataset, integration with APIs like Spotify for real-time and personalized music streaming is still in progress. Similarly, visual rendering logic has been planned and partially implemented using cached mood-based visuals, but it awaits real-time dynamic adjustments and AI-based generation.

The summary generation component also saw partial progress. A functional summarization model has been selected, but current results do not offer sufficiently structured and semantically smooth outputs. The team is still exploring better-performing models to achieve coherent summaries aligned with the narrative flow of the book.

Current Status Summary:

- **User Interface:** Almost finalized with responsive layout, key user features, and navigation
- **Emotion Detection:** Model selection and integration to emotion selection completed.
- **Music Recommendation:** Basic version implemented; Spotify API and personalization pending
- **Visual Generation:** Static visuals working; AI-based visuals in progress
- **Summarization:** Model integrated, but still searching for higher-quality alternatives
- **Final Backend Integration & Deployment:** In progress; includes API coordination, offline caching, and Docker-based containerization

Remaining Tasks and Future Improvements

As the project moves into the second term, the following major development and refinement tasks are planned:

- **Full Backend Integration:** Completion of the backend modules (Flask/Django), including secure API endpoints, database logic, and cloud deployment setup (e.g., AWS or Azure). This will enable full user authentication, session persistence, and cross-device synchronization.
- **Music Streaming Integration and Emotion-Based Matching:**
  music dataset categorized by genre and mood. In case the Spotify API is unavailable or deliberately excluded, the system will rely on this static dataset to continue emotion-based music matching. Both sources will support emotion-to-music mapping logic, ensuring a consistent auditory experience. This dual approach enhances system reliability and provides flexibility in music delivery, particularly in offline or limited-access environments.
- **Improved Summarization Pipeline:** Experimenting with extractive or hybrid summarization models to produce concise, coherent summaries across chapters. Custom fine-tuning and sentence alignment will also be evaluated.
- **Advanced Visual Adaptation:** Integration of AI-generated visuals (e.g., using Stable Diffusion or similar tools) for high-quality emotional feedback. Visual brightness and intensity will also be adapted in real-time.

- **Performance Optimization & Offline Mode:** Reducing latency in NLP processing for large books by preprocessing, segmentation, and caching. Also, expanding support for offline book, music, and visual access.
- **Testing & User Feedback:** Completion of beta testing, followed by extensive user acceptance testing (UAT) to validate the emotional experience and UI intuitiveness. Feedback from this process will guide final polish tasks.
- **Recommendation Engine Integration:** If time allows, a machine learning-based recommendation system will be incorporated, which tailors book and music suggestions based on user behavior and mood patterns.

10. References

- Hugging Face Transformers Documentation: https://huggingface.co/docs/transformers
- MongoDB Documentation: https://www.mongodb.com/docs/
- React Documentation: https://reactjs.org/docs/getting-started.html
- React Native Documentation: https://reactnative.dev/docs/getting-started
- pytest Documentation – https://docs.pytest.org/
- Jest Documentation – https://jestjs.io/docs/getting-started
- Postman API Platform – https://www.postman.com/
- Spotify API Documentation: https://developer.spotify.com/documentation/web-api
- IEEE Code of Ethics: https://www.ieee.org/about/corporate/governance/p7-8.html