Group 09

PageRhythm
Software Architecture Document

Version 1.1

Revision History

Date	Version	Description	Author
15/11/2024	1.0	The initial version of Software Architecture Document	Group 09
09/12/2024	1.1	- The implementation view of the project was updated to reflect changes in organization and folder structure of “FrontEnd” - More detailed explanations were provided to describe the nodes depicted in the deployment diagram. - The ER diagram and the relational database schema were updated to introduce some new attributes of some entities (“Creation Time” attribute of “Comment” and “Upload Time” of “Sample Audio File”) and reflect the changes in the domain of the attributes that act as primary key of some entities.	Nguyen Hoang Minh

Table of Contents

Software Architecture Document

1. Introduction

This document outlines the overall architectural design and structure of the PageRhythm web application. It serves as a blueprint for the system's development and fosters a shared understanding among all stakeholders.

a. Purposes

The document is written with the following purposes:

- System Overview: The document can offer a clear and concise description of the system's architecture for all stakeholders, including developers, testers, project managers, and customers.
- Development Reference: The document can serve as a reference for the development team, ensuring consistent design and implementation of all components.
- Change Impact Assessment: The document can provide a framework for assessing the impact of proposed changes on the system's architecture.
- Maintenance and Troubleshooting: The document can provide a clear overview that aids in the system's maintenance, making it easier to identify issues, debug, and optimize components.

b. Scopes

This Software Architecture Document outlines the overall architecture for the PageRhythm project, including its goals and design principles. It provides a detailed description of the architecture's components, the system's deployment strategy, and the implementation view specific to this project.

c. Acronyms, Abbreviations, and Definitions

Term	Explanation	Definition
HTTP	HyperText Transfer Protocol	HTTP is a protocol that defines rules for how messages are formatted and transmitted, and how web servers and browsers respond to various commands.
ER model	Entity-Relationship model	The ER model is a conceptual framework used to design and represent the structure of a database. It visually depicts the entities in the system, their attributes, and the relationships between them.

d. References

There are following available references:

- What is three-tier architecture? - IBM
- Lecture notes for the CS300 - Elements of Software Engineering course, Assoc. Prof. Nguyen Van Vu, 2024, VNU-HCM
- Supabase
- Render

e. Overview

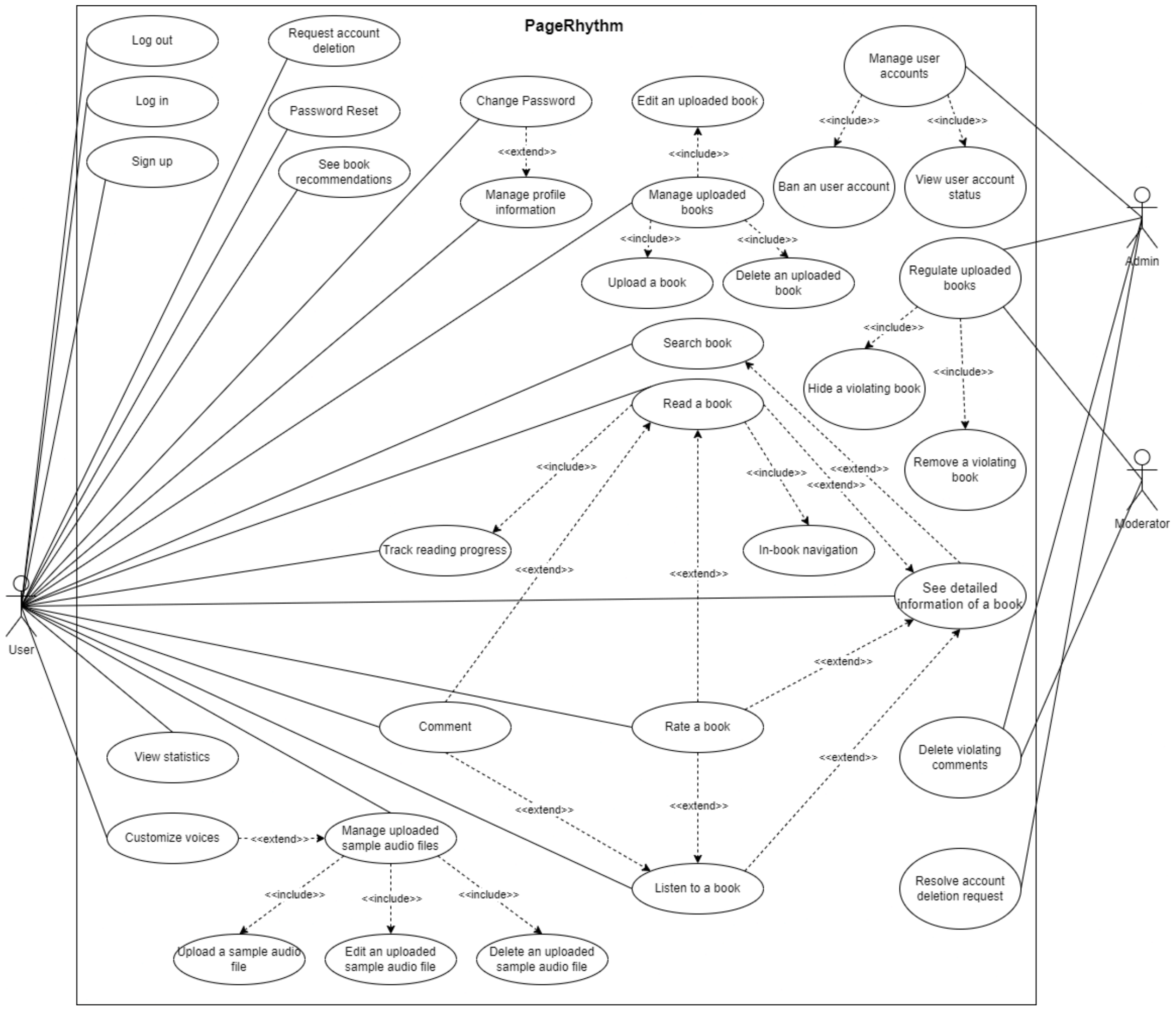
This Software Architecture Document is organized into the following main sections:

- Introduction: This section provides an overview of the entire Software Architecture Document, including the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the Software Architecture Document.
- Architectural Goals and Constraints: This section outlines the software requirements and objectives that significantly influence the architecture. It also highlights any special constraints, such as the design and implementation strategies or the development tools used.
- Use-Case Model:This section includes use-case diagrams that illustrate the functionality and interactions of the system.
- Logical View: This section presents the architecture, detailing the components and their relationships. A diagram depicting the architecture is included, along with descriptions of each component's responsibilities and the services it provides to other components. Additionally, the means of communication between components are explained.
- Deployment: This section describes how the system is deployed, including the mapping of components to the machines where they are executed.
- Implementation View:This section outlines the folder structures for the project code, corresponding to the components described in the Logical View section.

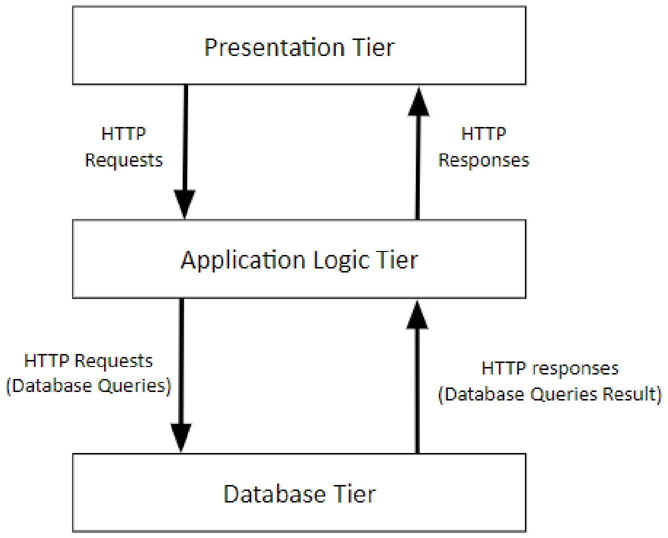
2. Architectural Goals and Constraints

- The architecture must follow the 3-tier architecture style.
- The architecture must ensure high maintainability by utilizing fine-grained, modular, and replaceable components. This design enables individual components to be updated or upgraded with minimal disruption to the rest of the system, ensuring seamless adaptability and long-term sustainability.
- The architecture must ensure high performance by enabling low-latency communication between tiers, optimizing data flow, and minimizing processing delays across the system. Communication between tiers should be efficient and kept to a necessary minimum to avoid overhead and ensure responsiveness.
- The architecture must ensure robust security by implementing strong authentication and authorization mechanisms within the business logic tier and safeguarding sensitive data in the database tier through access controls and other protective measures.
- The architecture must ensure a high level of privacy by maintaining the confidentiality of user data and implementing robust protective measures to prevent unauthorized access.
- The architecture must provide good reliability by ensuring consistent performance under varying conditions, allowing users to use the website without interruption.
- The architecture must guarantee safety.
- The architecture must ensure high availability to guarantee that the system remains operational and accessible to users at all times.
- The architecture is currently designed exclusively to support web-based platforms.
- The front-end (the presentation tier) is developed using Typescript and the React library.
- The back-end (the application logic tier) is developed using Python and the Flask framework, which is a micro web framework.
- The database is developed using PostgreSQL.

3. Use-Case Model



4. Logical View



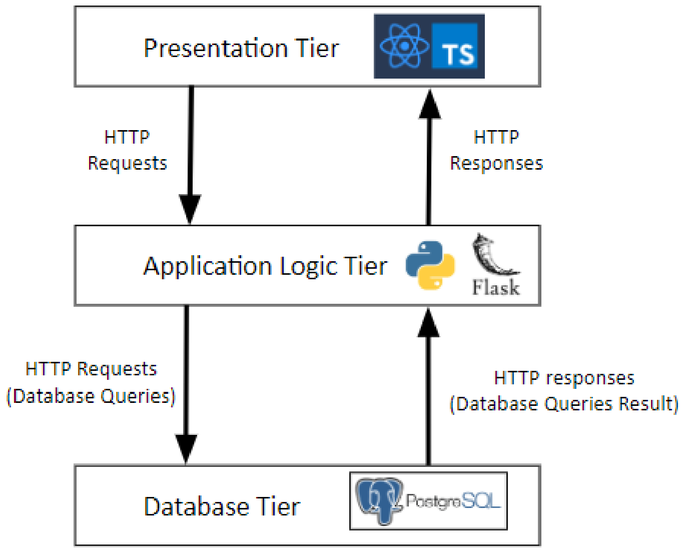
The application developed by our team follows a 3-tier architecture, with the system divided into three distinct layers, as illustrated in the diagram above:

- **Presentation Tier**:
  - This layer is responsible for the user interface and user experience. It handles the display of data to the user and captures user inputs, which are then passed to the application logic tier for processing.
  - Our group uses the Typescript programming language and the React library to implement this tier.
- **Application Logic Tier**:
  - This tier might be also called the business logic or service layer. The tier contains the core functionality of the application. It processes the data received from the presentation tier, interacts with the database tier as needed, and applies business rules to fulfill the user requests.
  - Our group uses the Python programming language and the Flask micro web framework to implement this tier.
- **Database Tier**:
  - This layer is responsible for data storage and management. It handles the retrieval, insertion, update, and deletion of data in the database, ensuring the persistence and integrity of the application's data.
  - Our group uses PostgreSQL to implement this tier.

Each tier is designed to operate independently, which enhances the scalability, maintainability, and flexibility of the system.

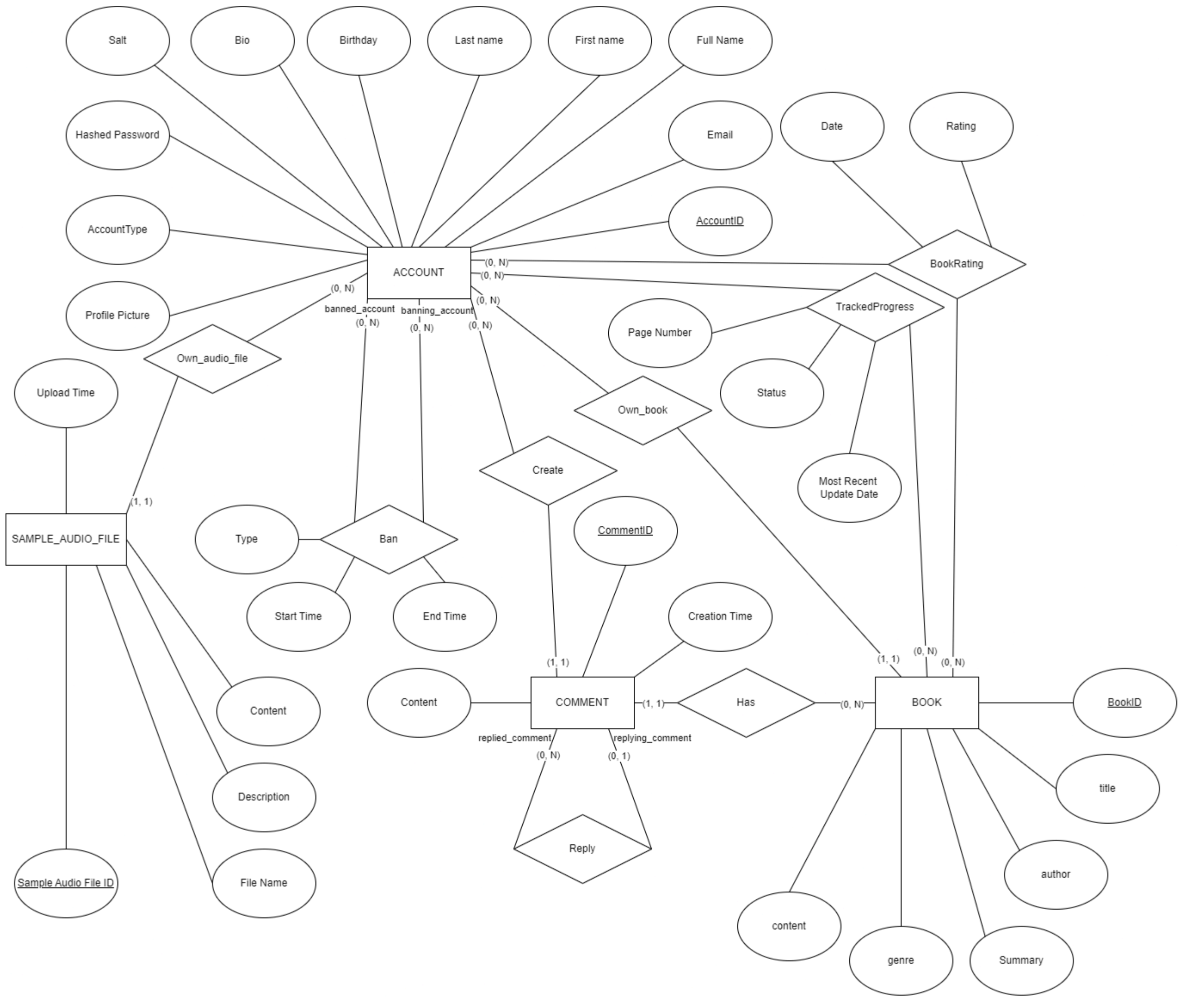
The diagram also illustrates that HTTP requests and responses are used for communication between the tiers, enabling seamless interaction across the system.

The following diagram builds upon the first diagram in this section, highlighting the languages and tools our team used to develop each specific tier of the system.

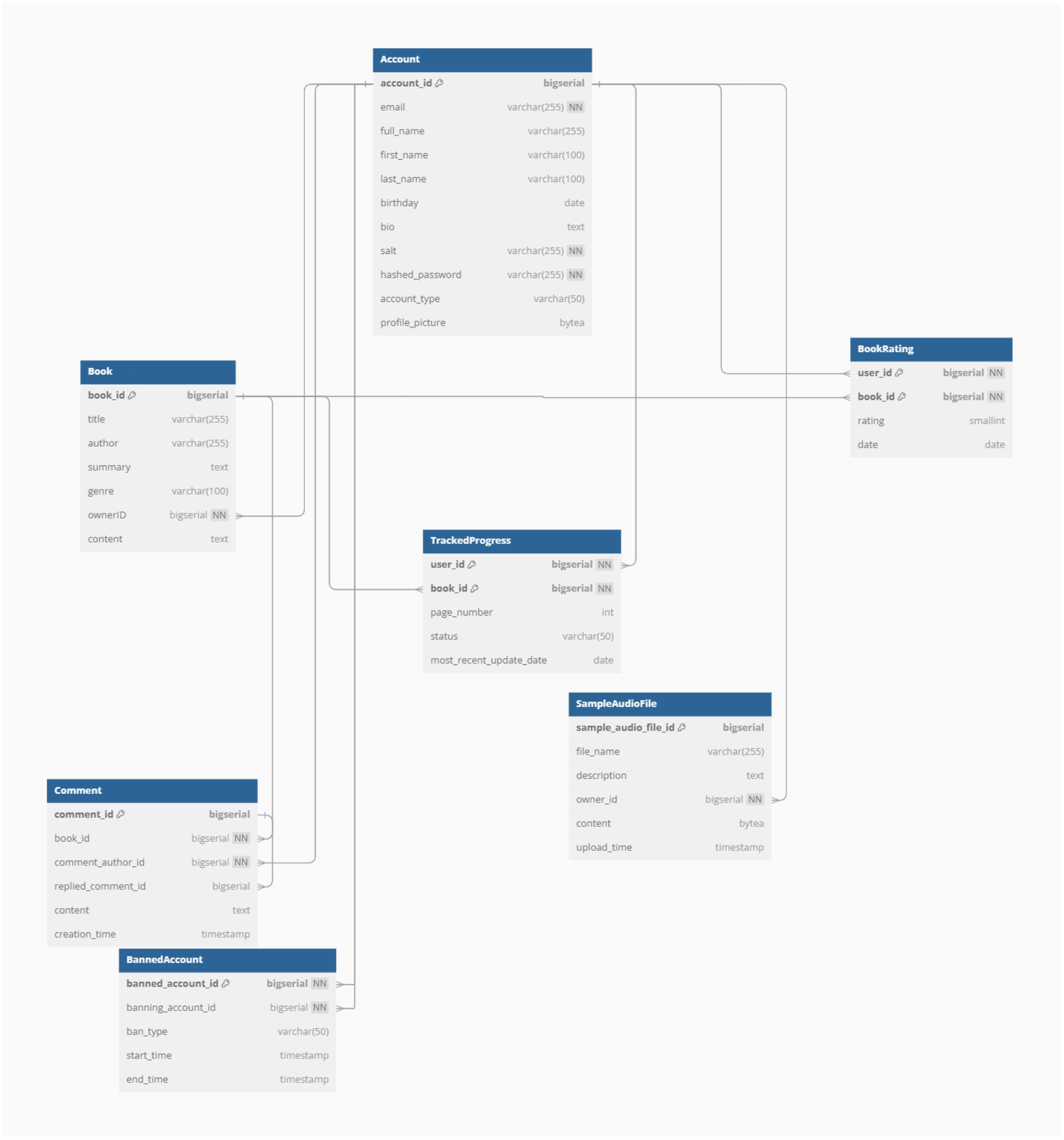


4.1 Component: Database Tier

The Entity-Relationship (ER) models for our application are illustrated in the diagram below:



- PostgreSQL is used to implement this component.
- Based on the requirements documented, the above diagram illustrates the Entity-Relationship models for the database of the application.
- The Entity-Relationship models include following entities:
    - ACCOUNT
    - SAMPLE_AUDIO_FILE
    - COMMENT
    - BOOK
- The Entity-Relationship models include following relationships:
    - BookRating
    - TrackedProgress
    - Own_book
    - Create
    - Ban
    - Reply
    - Own_audio_file
- The following diagram describes the used relational database schema.



Account	
account_id	bigserial
email	varchar(255) NN
full_name	varchar(255)
first_name	varchar(100)
last_name	varchar(100)
birthday	date
bio	text
salt	varchar(255) NN
hashed_password	varchar(255) NN
account_type	varchar(50)
profile_picture	bytea

Book	
book_id	bigserial
title	varchar(255)
author	varchar(255)
summary	text
genre	varchar(100)
ownerID	bigserial NN
content	text

BookRating	
user_id	bigserial NN
book_id	bigserial NN
rating	smallint
date	date

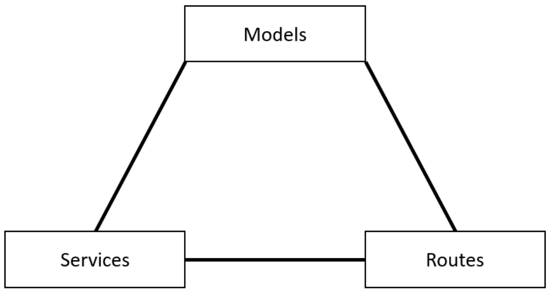
TrackedProgress	
user_id	bigserial NN
book_id	bigserial NN
page_number	int
status	varchar(50)
most_recent_update_date	date

SampleAudioFile	
sample_audio_file_id	bigserial
file_name	varchar(255)
description	text
owner_id	bigserial NN
content	bytea
upload_time	timestamp

Comment	
comment_id	bigserial
book_id	bigserial NN
comment_author_id	bigserial NN
replied_comment_id	bigserial
content	text
creation_time	timestamp

BannedAccount	
banned_account_id	bigserial NN
banning_account_id	bigserial NN
ban_type	varchar(50)
start_time	timestamp
end_time	timestamp

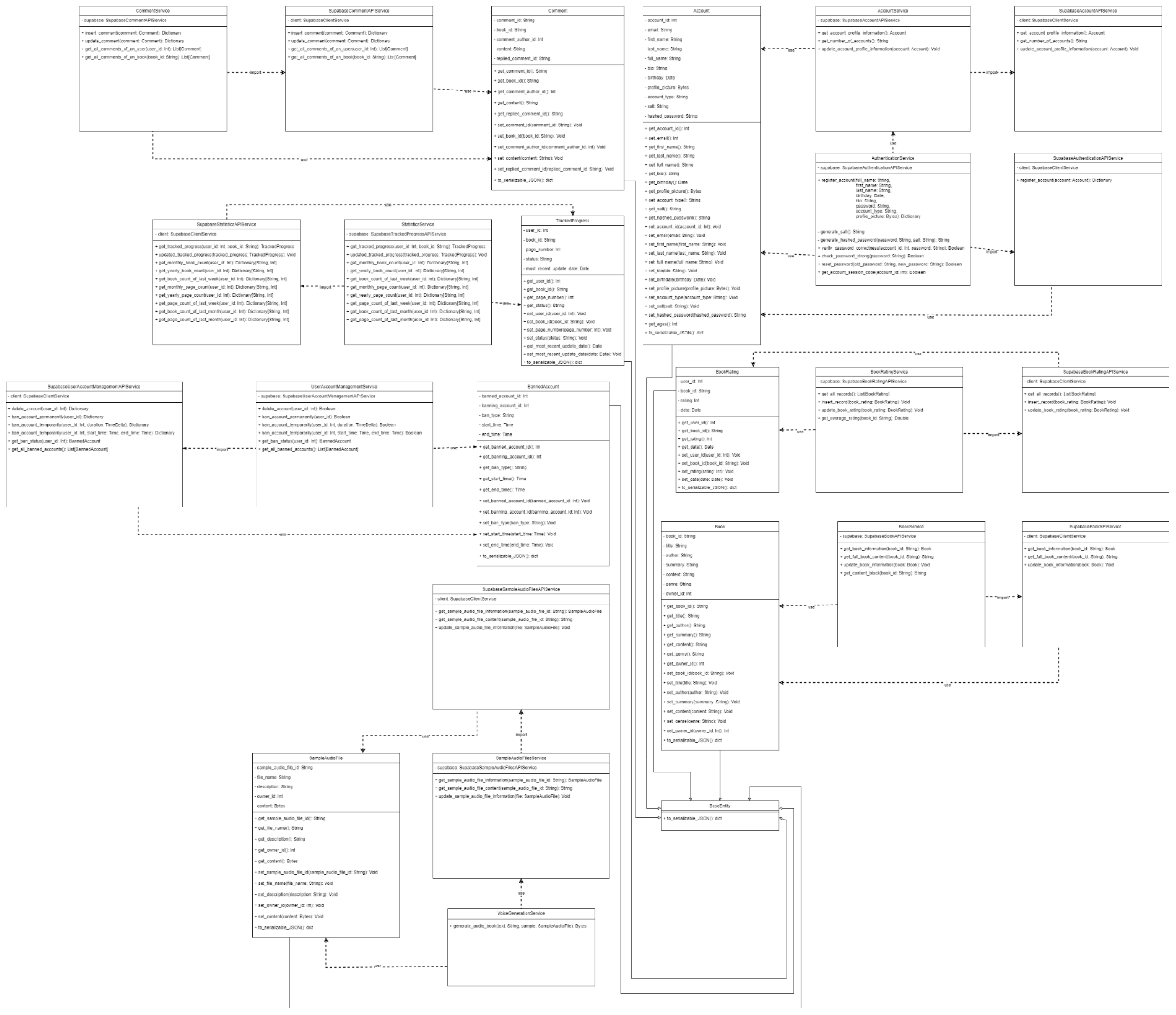
4.2 Component: Application Logic Tier



This component can be further subdivided into three distinct sub-components, each serving a unique role in the overall architecture:

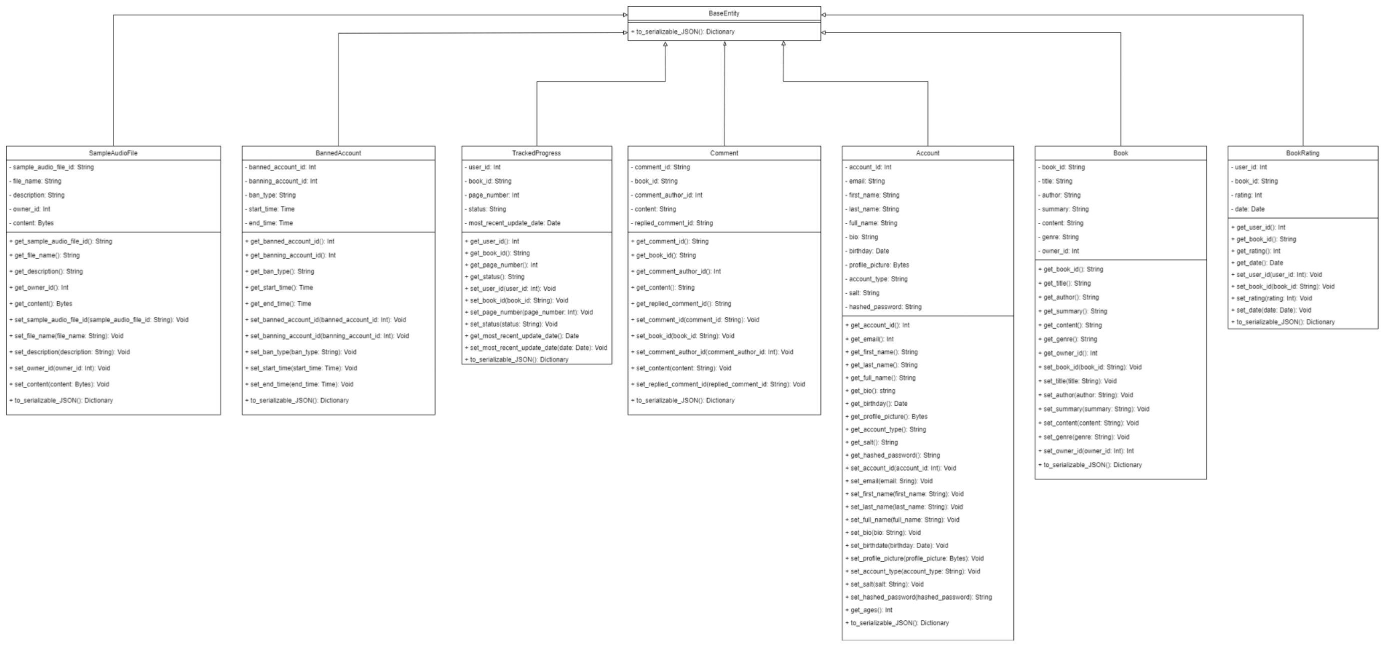
- Models: This sub-component is responsible for defining the data structures and entities required for interacting with the database or handling data within the application. It serves as the foundation for data representation and provides an abstraction layer for the data used throughout the system.
- Services: The services sub-component encapsulates the core business logic of the application. It defines the methods and processes that manipulate the data, interact with external APIs, and provide functionality to other components. Services are the intermediary between the models (which define the data structure) and the routes (which handle user interactions).
- Routes: This sub-component is responsible for defining the API endpoints that expose the application's functionality to external clients. It acts as the interface between the client-facing presentation tier and the application's backend logic. Routes handle incoming requests, interact with services, and return appropriate responses.

The Python programming language and the Flask micro web framework is used to implement this component.

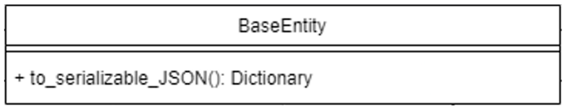


CommentService
SupabaseCommentAPIService
Comment
Account
AccountService
SupabaseAccountAPIService
AuthenticationService
SupabaseAuthenticationAPIService
SupabaseStatisticsAPIService
StatisticsService
TrackedProgress
SupabaseUserAccountManagementAPIService
UserAccountManagementService
BannedAccount
BookRating
BookRatingService
SupabaseBookRatingAPIService
Book
BookService
SupabaseBookAPIService
SupabaseSampleAudioFilesAPIService
SampleAudioFile
SampleAudioFilesService
VoiceGenerationService
BaseEntity
import
use

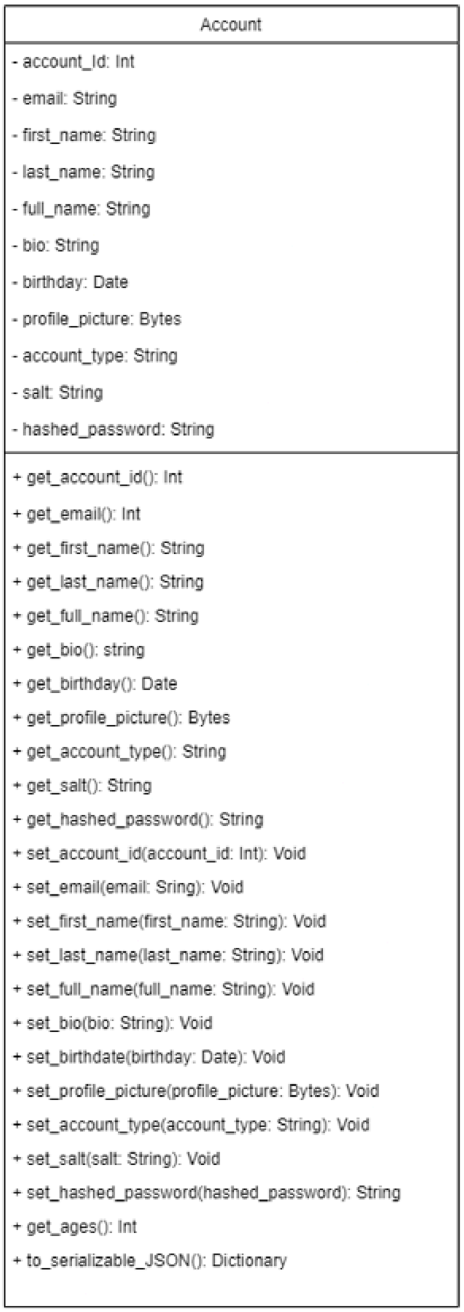
4.2.1 Models


BaseEntity
+ to_serializable_JSON(): Dictionary
SampleAudioFile
- sample_audio_file_id: String
- file_name: String
- description: String
- owner_id: Int
- content: Bytes
+ get_sample_audio_file_id(): String
+ get_file_name(): String
+ get_description(): String
+ get_owner_id(): Int
+ get_content(): Bytes
+ set_sample_audio_file_id(sample_audio_file_id: String): Void
+ set_file_name(file_name: String): Void
+ set_description(description: String): Void
+ set_owner_id(owner_id: Int): Void
+ set_content(content: Bytes): Void
+ to_serializable_JSON(): Dictionary
BannedAccount
- banned_account_id: Int
- banning_account_id: Int
- ban_type: String
- start_time: Time
- end_time: Time
+ get_banned_account_id(): Int
+ get_banning_account_id(): Int
+ get_ban_type(): String
+ get_start_time(): Time
+ get_end_time(): Time
+ set_banned_account_id(banned_account_id: Int): Void
+ set_banning_account_id(banning_account_id: Int): Void
+ set_ban_type(ban_type: String): Void
+ set_start_time(start_time: Time): Void
+ set_end_time(end_time: Time): Void
+ to_serializable_JSON(): Dictionary
TrackedProgress
- user_id: Int
- book_id: String
- page_number: Int
- status: String
- most_recent_update_date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_page_number(): Int
+ get_status(): String
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_page_number(page_number: Int): Void
+ set_status(status: String): Void
+ get_most_recent_update_date(): Date
+ set_most_recent_update_date(date: Date): Void
+ to_serializable_JSON(): Dictionary
Comment
- comment_id: String
- book_id: String
- comment_author_id: Int
- content: String
- replied_comment_id: String
+ get_comment_id(): String
+ get_book_id(): String
+ get_comment_author_id(): Int
+ get_content(): String
+ get_replied_comment_id(): String
+ set_comment_id(comment_id: String): Void
+ set_book_id(book_id: String): Void
+ set_comment_author_id(comment_author_id: Int): Void
+ set_content(content: String): Void
+ set_replied_comment_id(replied_comment_id: String): Void
+ to_serializable_JSON(): Dictionary
Account
- account_Id: Int
- email: String
- first_name: String
- last_name: String
- full_name: String
- bio: String
- birthday: Date
- profile_picture: Bytes
- account_type: String
- salt: String
- hashed_password: String
+ get_account_id(): Int
+ get_email(): Int
+ get_first_name(): String
+ get_last_name(): String
+ get_full_name(): String
+ get_bio(): string
+ get_birthday(): Date
+ get_profile_picture(): Bytes
+ get_account_type(): String
+ get_salt(): String
+ get_hashed_password(): String
+ set_account_id(account_id: Int): Void
+ set_email(email: Sring): Void
+ set_first_name(first_name: String): Void
+ set_last_name(last_name: String): Void
+ set_full_name(full_name: String): Void
+ set_bio(bio: String): Void
+ set_birthdate(birthday: Date): Void
+ set_profile_picture(profile_picture: Bytes): Void
+ set_account_type(account_type: String): Void
+ set_salt(salt: String): Void
+ set_hashed_password(hashed_password): String
+ get_ages(): Int
+ to_serializable_JSON(): Dictionary
Book
- book_id: String
- title: String
- author: String
- summary: String
- content: String
- genre: String
- owner_id: Int
+ get_book_id(): String
+ get_title(): String
+ get_author(): String
+ get_summary(): String
+ get_content(): String
+ get_genre(): String
+ get_owner_id(): Int
+ set_book_id(book_id: String): Void
+ set_title(title: String): Void
+ set_author(author: String): Void
+ set_summary(summary: String): Void
+ set_content(content: String): Void
+ set_genre(genre: String): Void
+ set_owner_id(owner_id: Int): Int
+ to_serializable_JSON(): Dictionary
BookRating
- user_id: Int
- book_id: String
- rating: Int
- date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_rating(): Int
+ get_date(): Date
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_rating(rating: Int): Void
+ set_date(date: Date): Void
+ to_serializable_JSON(): Dictionary

All concrete classes representing the models used in the application inherit from a single base class named BaseEntity. This base class provides a single utility function designed to convert the data stored within its objects into a JSON-serializable structure.



4.2.1.1 Account

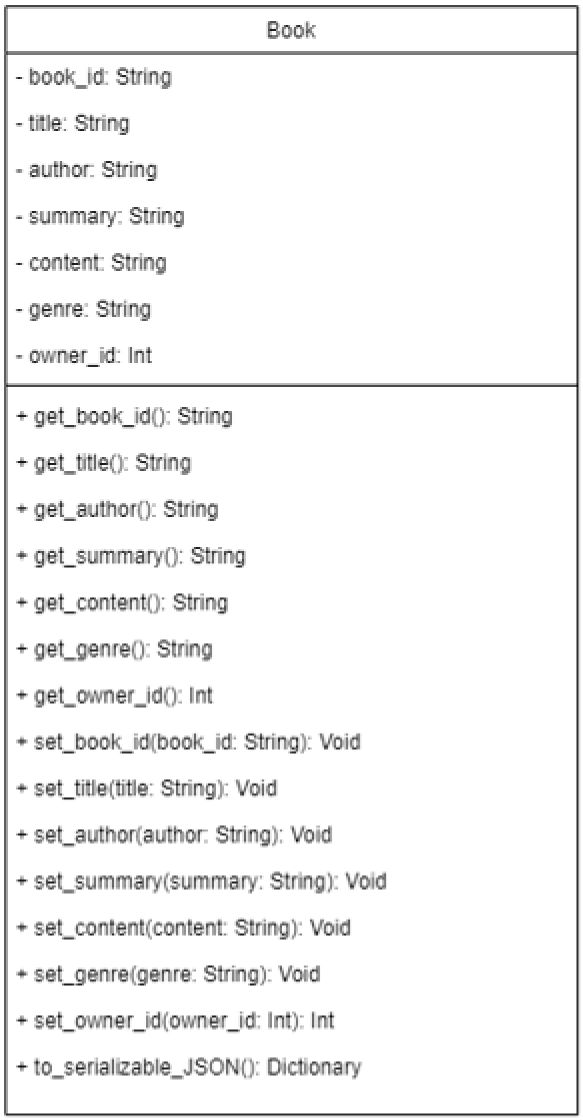


Account
- account_Id: Int
- email: String
- first_name: String
- last_name: String
- full_name: String
- bio: String
- birthday: Date
- profile_picture: Bytes
- account_type: String
- salt: String
- hashed_password: String
+ get_account_id(): Int
+ get_email(): Int
+ get_first_name(): String
+ get_last_name(): String
+ get_full_name(): String
+ get_bio(): string
+ get_birthday(): Date
+ get_profile_picture(): Bytes
+ get_account_type(): String
+ get_salt(): String
+ get_hashed_password(): String
+ set_account_id(account_id: Int): Void
+ set_email(email: Sring): Void
+ set_first_name(first_name: String): Void
+ set_last_name(last_name: String): Void
+ set_full_name(full_name: String): Void
+ set_bio(bio: String): Void
+ set_birthdate(birthday: Date): Void
+ set_profile_picture(profile_picture: Bytes): Void
+ set_account_type(account_type: String): Void
+ set_salt(salt: String): Void
+ set_hashed_password(hashed_password): String
+ get_ages(): Int
+ to_serializable_JSON(): Dictionary

Account: This class represents a user account, encompassing both profile details and authentication-related information. Additionally, it provides methods for retrieving and updating the account's attributes, ensuring seamless management of user data.

- **get_account_id**: Return the ID of the account.
- **get_email**: Return the email of the account.
- **get_first_name**: Return the first name registered for the account.
- **get_last_name**: Return the last name registered for the account.
- **get_full_name**: Return the full name registered for the account.
- **get_bio**: Return the biography associated with the account.
- **get_birthday**: Return the date of birth of the account holder.
- **get_profile_picture**: Return the profile picture associated with the account.
- **get_account_type**: Return the type of account.
- **get_salt**: Return the salt used for password hashing.
- **get_hashed_password**: Return the hashed password for the account.
- **set_account_id**: Set the ID of the account.
- **set_email**: Set the email for the account.
- **set_first_name**: Set the first name for the account.
- **set_last_name**: Set the last name for the account.
- **set_full_name**: Set the full name for the account.
- **set_bio**: Set the biography for the account.
- **set_birthday**: Set the date of birth for the account holder.
- **set_profile_picture**: Set the profile picture for the account.
- **set_account_type**: Set the type of account.
- **set_salt**: Set the salt used for password hashing.
- **set_hashed_password**: Set the hashed password for the account.
- **get_age**: Return the age of the account holder based on the registered birthday.

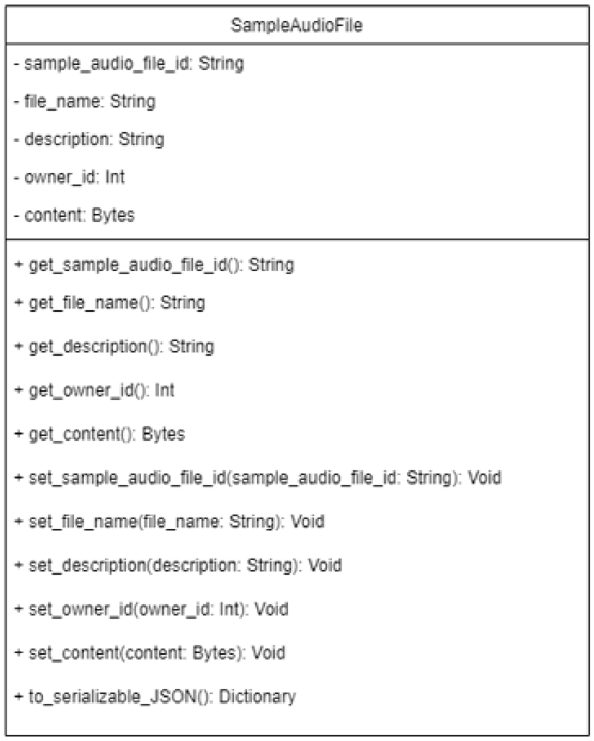
4.2.1.2 Book


Book
- book_id: String
- title: String
- author: String
- summary: String
- content: String
- genre: String
- owner_id: Int
+ get_book_id(): String
+ get_title(): String
+ get_author(): String
+ get_summary(): String
+ get_content(): String
+ get_genre(): String
+ get_owner_id(): Int
+ set_book_id(book_id: String): Void
+ set_title(title: String): Void
+ set_author(author: String): Void
+ set_summary(summary: String): Void
+ set_content(content: String): Void
+ set_genre(genre: String): Void
+ set_owner_id(owner_id: Int): Int
+ to_serializable_JSON(): Dictionary

Book: This class represents a book, encompassing both book detailed information and its content. Additionally, it provides methods for retrieving and updating the book's attributes, ensuring seamless management of book data.

- **get_book_id**: Return the ID of the book.
- **get_title**: Return the title of the book.
- **get_author**: Return the author of the book.
- **get_summary**: Return the summary of the book.
- **get_content**: Return the content of the book.
- **get_genre**: Return the genre of the book.
- **get_owner_id**: Return the ID of the owner of the book.
- **set_book_id**: Set the ID of the book.
- **set_title**: Set the title of the book.
- **set_author**: Set the author of the book.
- **set_summary**: Set the summary of the book.
- **set_content**: Set the content of the book.
- **set_genre**: Set the genre of the book.
- **set_owner_id**: Set the ID of the owner of the book.

4.2.1.3 Sample Audio File

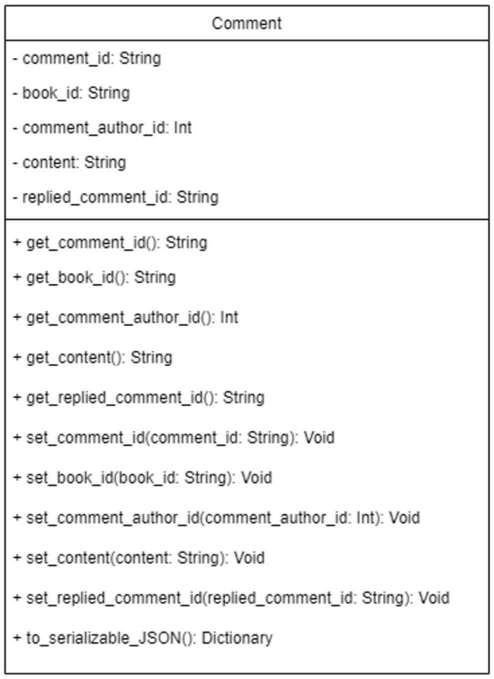


SampleAudioFile
- sample_audio_file_id: String
- file_name: String
- description: String
- owner_id: Int
- content: Bytes
+ get_sample_audio_file_id(): String
+ get_file_name(): String
+ get_description(): String
+ get_owner_id(): Int
+ get_content(): Bytes
+ set_sample_audio_file_id(sample_audio_file_id: String): Void
+ set_file_name(file_name: String): Void
+ set_description(description: String): Void
+ set_owner_id(owner_id: Int): Void
+ set_content(content: Bytes): Void
+ to_serializable_JSON(): Dictionary

Sample Audio File: This class represents a sample audio file, encompassing both information of the file and its content. Additionally, it provides methods for retrieving and updating the file's attributes, ensuring seamless management of sample audio files.

- **get_sample_audio_file_id**: Returns the ID of the sample audio file.
- **get_file_name**: Returns the file name of the sample audio file.
- **get_description**: Returns the description of the sample audio file.
- **get_owner_id**: Returns the ID of the owner of the sample audio file.
- **get_content**: Returns the content of the sample audio file.
- **set_sample_audio_file_id**: Sets the ID of the sample audio file.
- **set_file_name**: Sets the file name of the sample audio file.
- **set_description**: Sets the description of the sample audio file.
- **set_owner_id**: Sets the ID of the owner of the sample audio file.
- **set_content**: Sets the content of the sample audio file.

4.2.1.4 Comment



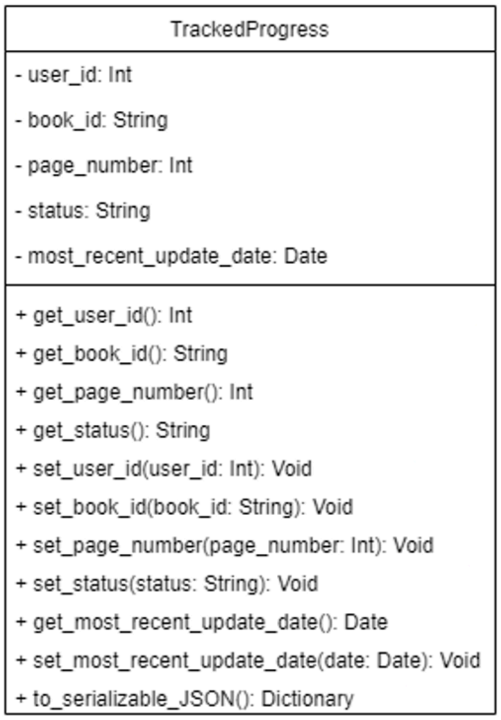
Comment
- comment_id: String
- book_id: String
- comment_author_id: Int
- content: String
- replied_comment_id: String
+ get_comment_id(): String
+ get_book_id(): String
+ get_comment_author_id(): Int
+ get_content(): String
+ get_replied_comment_id(): String
+ set_comment_id(comment_id: String): Void
+ set_book_id(book_id: String): Void
+ set_comment_author_id(comment_author_id: Int): Void
+ set_content(content: String): Void
+ set_replied_comment_id(replied_comment_id: String): Void
+ to_serializable_JSON(): Dictionary

Comment: This class represents a comment made by a user.

- **get_comment_id**: Return the ID of the comment.
- **get_book_id**: Return the ID of the book associated with the comment.
- **get_comment_author_id**: Return the ID of the author who wrote the comment.
- **get_content**: Return the content of the comment.
- **get_replied_comment_id**: Return the ID of the comment that this comment is replying to (if applicable).
- **set_comment_id**: Set the ID of the comment.
- **set_book_id**: Set the ID of the book associated with the comment.
- **set_comment_author_id**: Set the ID of the author who wrote the comment.
- **set_content**: Set the content of the

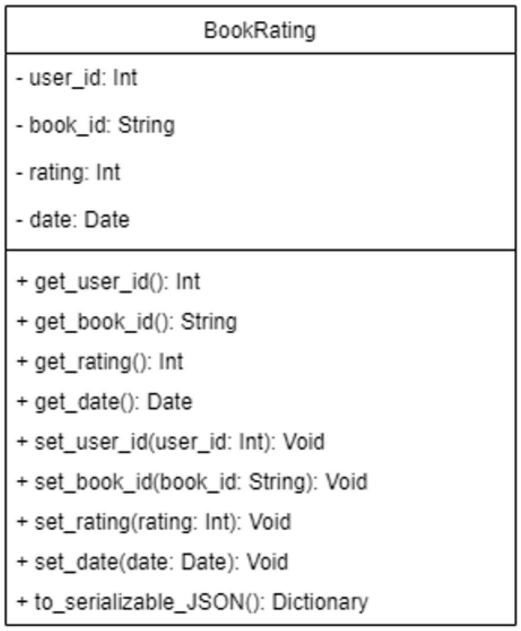
comment.
- **set_replied_comment_id**: Set the ID of the comment being replied to.

4.2.1.5 Tracked Progress


TrackedProgress
- user_id: Int
- book_id: String
- page_number: Int
- status: String
- most_recent_update_date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_page_number(): Int
+ get_status(): String
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_page_number(page_number: Int): Void
+ set_status(status: String): Void
+ get_most_recent_update_date(): Date
+ set_most_recent_update_date(date: Date): Void
+ to_serializable_JSON(): Dictionary

TrackedProgress: This class represents a user's progress in reading a book, tracking key details such as the current reading status, pages completed.
- **get_user_id**: Returns the ID of the user.
- **get_book_id**: Returns the ID of the book.
- **get_page_number**: Returns the current page number.
- **get_status**: Returns the current status of the book.
- **get_most_recent_update_date**: Returns the date of the most recent update to the tracked progress.
- **set_user_id**: Sets the ID of the user.
- **set_book_id**: Sets the ID of the book.
- **set_page_number**: Sets the current page number.
- **set_status**: Sets the current status of the book.
- **set_most_recent_update_date**: Sets the date of the most recent update to the tracked progress.

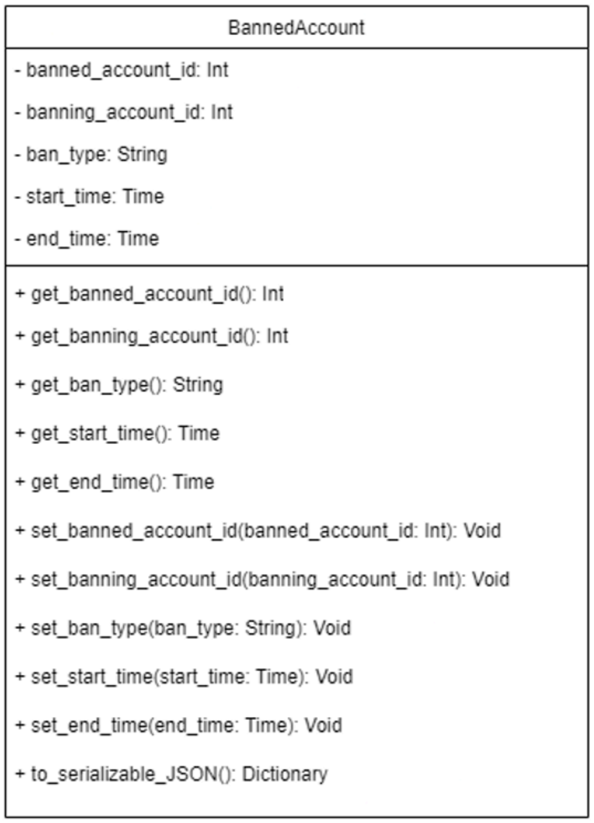
4.2.1.6 Book Rating


BookRating
- user_id: Int
- book_id: String
- rating: Int
- date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_rating(): Int
+ get_date(): Date
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_rating(rating: Int): Void
+ set_date(date: Date): Void
+ to_serializable_JSON(): Dictionary

BookRating: This class represents rating given by a specific user to a specific book.

- **get_user_id**: Returns the ID of the user who gave the rating.
- **get_book_id**: Returns the ID of the book that was rated.
- **get_rating**: Returns the rating given to the book.
- **get_date**: Returns the date the rating was given.
- **set_user_id**: Sets the ID of the user who gave the rating.
- **set_book_id**: Sets the ID of the book that was rated.
- **set_rating**: Sets the rating given to the book.
- **set_date**: Sets the date the rating was given.

4.2.1.7 Banned Account



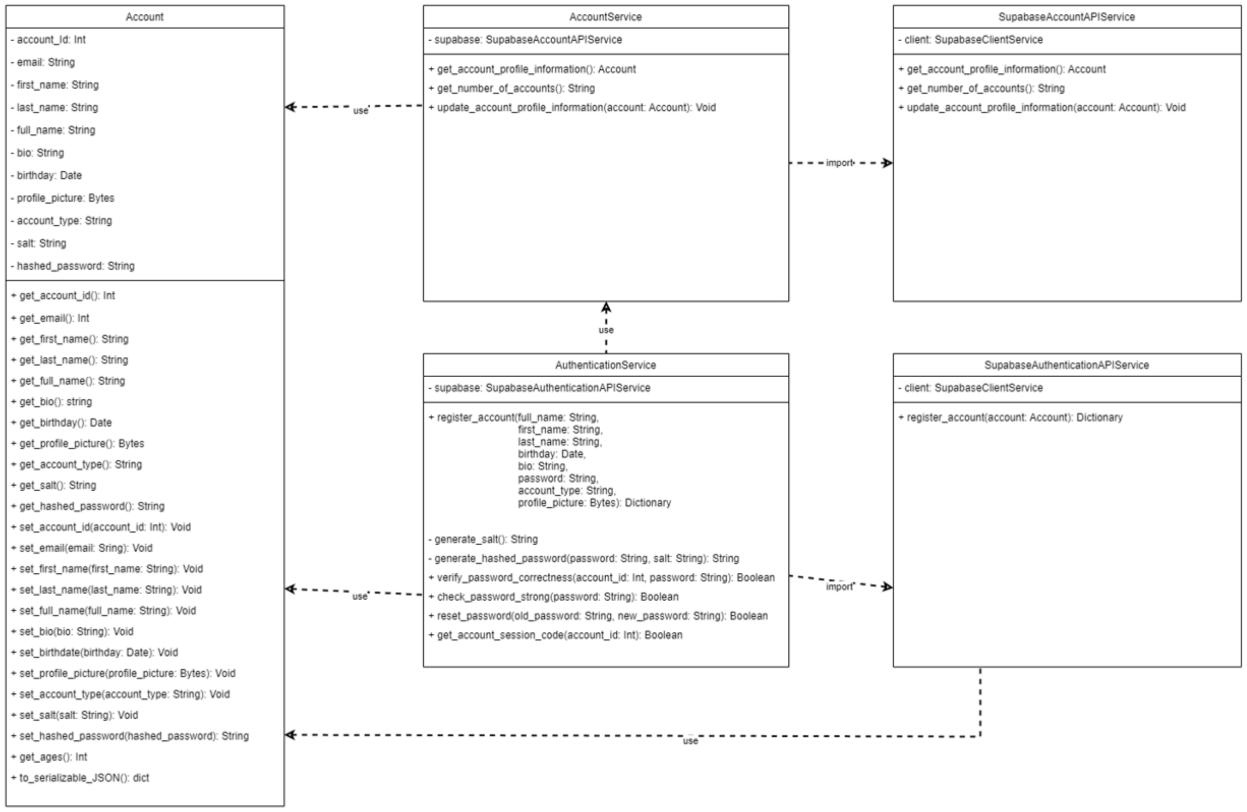
BannedAccount
- banned_account_id: Int
- banning_account_id: Int
- ban_type: String
- start_time: Time
- end_time: Time
+ get_banned_account_id(): Int
+ get_banning_account_id(): Int
+ get_ban_type(): String
+ get_start_time(): Time
+ get_end_time(): Time
+ set_banned_account_id(banned_account_id: Int): Void
+ set_banning_account_id(banning_account_id: Int): Void
+ set_ban_type(ban_type: String): Void
+ set_start_time(start_time: Time): Void
+ set_end_time(end_time: Time): Void
+ to_serializable_JSON(): Dictionary

BannedAccount: This class represents a ban on an account.

- **get_banned_account_id**: Returns the ID of the banned account.
- **get_banning_account_id**: Returns the ID of the account that banned the account.
- **get_ban_type**: Returns the type of ban.
- **get_start_time**: Returns the start time of the ban.
- **get_end_time**: Returns the end time of the ban.
- **set_banned_account_id**: Sets the ID of the banned account.
- **set_banning_account_id**: Sets the ID of the account that banned the account.
- **set_ban_type**: Sets the type of ban.
- **set_start_time**: Sets the start time of the ban.
- **set_end_time**: Sets the end time of the ban.

4.2.2 Services

4.2.2.1 Account Services



Account
- account_Id: Int
- email: String
- first_name: String
- last_name: String
- full_name: String
- bio: String
- birthday: Date
- profile_picture: Bytes
- account_type: String
- salt: String
- hashed_password: String
+ get_account_id(): Int
+ get_email(): Int
+ get_first_name(): String
+ get_last_name(): String
+ get_full_name(): String
+ get_bio(): string
+ get_birthday(): Date
+ get_profile_picture(): Bytes
+ get_account_type(): String
+ get_salt(): String
+ get_hashed_password(): String
+ set_account_id(account_id: Int): Void
+ set_email(email: Sring): Void
+ set_first_name(first_name: String): Void
+ set_last_name(last_name: String): Void
+ set_full_name(full_name: String): Void
+ set_bio(bio: String): Void
+ set_birthdate(birthday: Date): Void
+ set_profile_picture(profile_picture: Bytes): Void
+ set_account_type(account_type: String): Void
+ set_salt(salt: String): Void
+ set_hashed_password(hashed_password): String
+ get_ages(): Int
+ to_serializable_JSON(): dict
AccountService
- supabase: SupabaseAccountAPIService
+ get_account_profile_information(): Account
+ get_number_of_accounts(): String
+ update_account_profile_information(account: Account): Void
SupabaseAccountAPIService
- client: SupabaseClientService
+ get_account_profile_information(): Account
+ get_number_of_accounts(): String
+ update_account_profile_information(account: Account): Void
AuthenticationService
- supabase: SupabaseAuthenticationAPIService
+ register_account(full_name: String, first_name: String, last_name: String, birthday: Date, bio: String, password: String, account_type: String, profile_picture: Bytes): Dictionary
- generate_salt(): String
- generate_hashed_password(password: String, salt: String): String
+ verify_password_correctness(account_id: Int, password: String): Boolean
+ check_password_strong(password: String): Boolean
+ reset_password(old_password: String, new_password: String): Boolean
+ get_account_session_code(account_id: Int): Boolean
SupabaseAuthenticationAPIService
- client: SupabaseClientService
+ register_account(account: Account): Dictionary
use
use
use
use
import
import

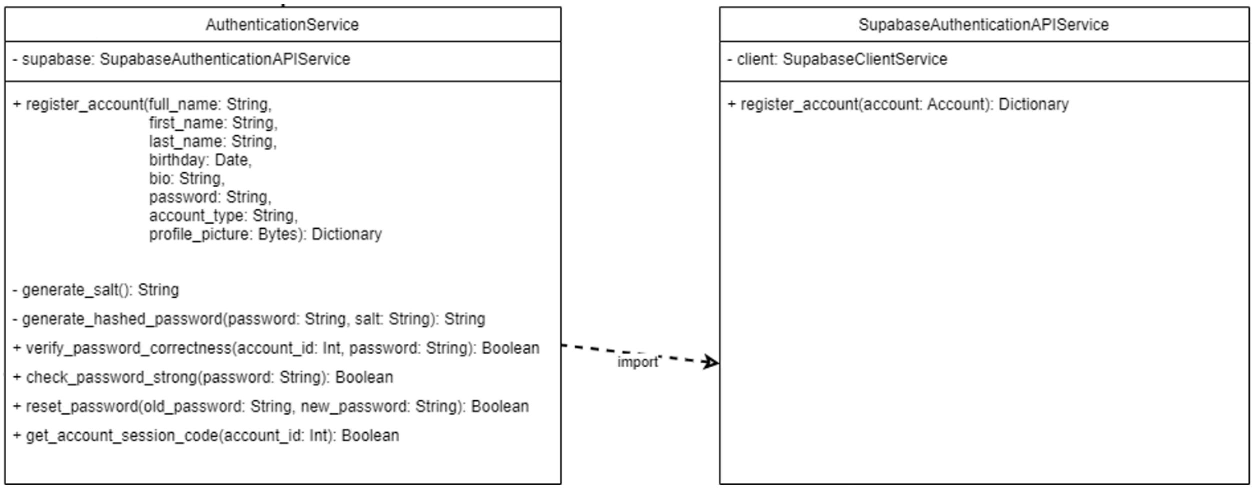
AccountService: This class provides various functionalities related to managing accounts information, serving as the primary interface for account-related operations.

- **get_account_profile_information**: Returns the profile information of an account.
- **get_number_of_accounts**: Returns the total number of accounts.
- **update_account_profile_information**: Updates the profile information of an account.

SupabaseAccountAPIService: This class is responsible for implementing methods that enable the AccountService to interact with the database seamlessly. It acts as a bridge between the "AccountService" and the underlying database.

- **get_account_profile_information:** Returns the profile information of an account stored in the database.
- **get_number_of_accounts:** Returns the total number of accounts stored in the database.
- **update_account_profile_information:** Updates the profile information of an account stored in the database.

4.2.2.2 Authentication Services


AuthenticationService
- supabase: SupabaseAuthenticationAPIService
+ register_account(full_name: String, first_name: String, last_name: String, birthday: Date, bio: String, password: String, account_type: String, profile_picture: Bytes): Dictionary
- generate_salt(): String
- generate_hashed_password(password: String, salt: String): String
+ verify_password_correctness(account_id: Int, password: String): Boolean
+ check_password_strong(password: String): Boolean
+ reset_password(old_password: String, new_password: String): Boolean
+ get_account_session_code(account_id: Int): Boolean
import
SupabaseAuthenticationAPIService
- client: SupabaseClientService
+ register_account(account: Account): Dictionary

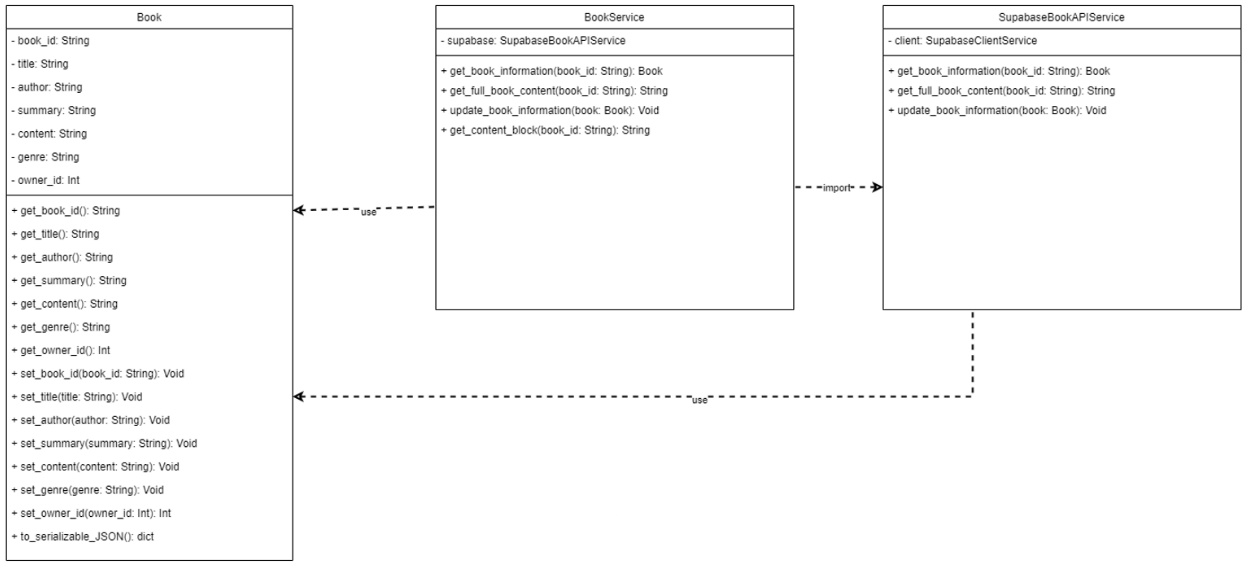
AuthenticationService: This class provides various functionalities related to managing user authentication, including registration, password management, and session management.

- **register_account:** Registers a new account with the provided information.
- **generate_salt:** Generates a new salt for password hashing.
- **generate_hashed_password:** Generates a hashed password using the given password and salt.
- **verify_password_correctness:** Verifies the correctness of a provided password for a given account.
- **check_password_strong:** Checks the strength of a password.
- **reset_password:** Resets the password for an account.
- **get_account_session_code:** Generates a session code for an account.

SupabaseAuthenticationAPIService: This class is responsible for implementing methods that enable the AuthenticationService to interact with the Supabase database seamlessly. It acts as a bridge between the "AuthenticationService" and the underlying Supabase database.

- **register_account:** Registers a new account in the Supabase database.

4.2.2.3 Book Services



Book
- book_id: String
- title: String
- author: String
- summary: String
- content: String
- genre: String
- owner_id: Int
+ get_book_id(): String
+ get_title(): String
+ get_author(): String
+ get_summary(): String
+ get_content(): String
+ get_genre(): String
+ get_owner_id(): Int
+ set_book_id(book_id: String): Void
+ set_title(title: String): Void
+ set_author(author: String): Void
+ set_summary(summary: String): Void
+ set_content(content: String): Void
+ set_genre(genre: String): Void
+ set_owner_id(owner_id: Int): Int
+ to_serializable_JSON(): dict
BookService
- supabase: SupabaseBookAPIService
+ get_book_information(book_id: String): Book
+ get_full_book_content(book_id: String): String
+ update_book_information(book: Book): Void
+ get_content_block(book_id: String): String
SupabaseBookAPIService
- client: SupabaseClientService
+ get_book_information(book_id: String): Book
+ get_full_book_content(book_id: String): String
+ update_book_information(book: Book): Void
use
import
use

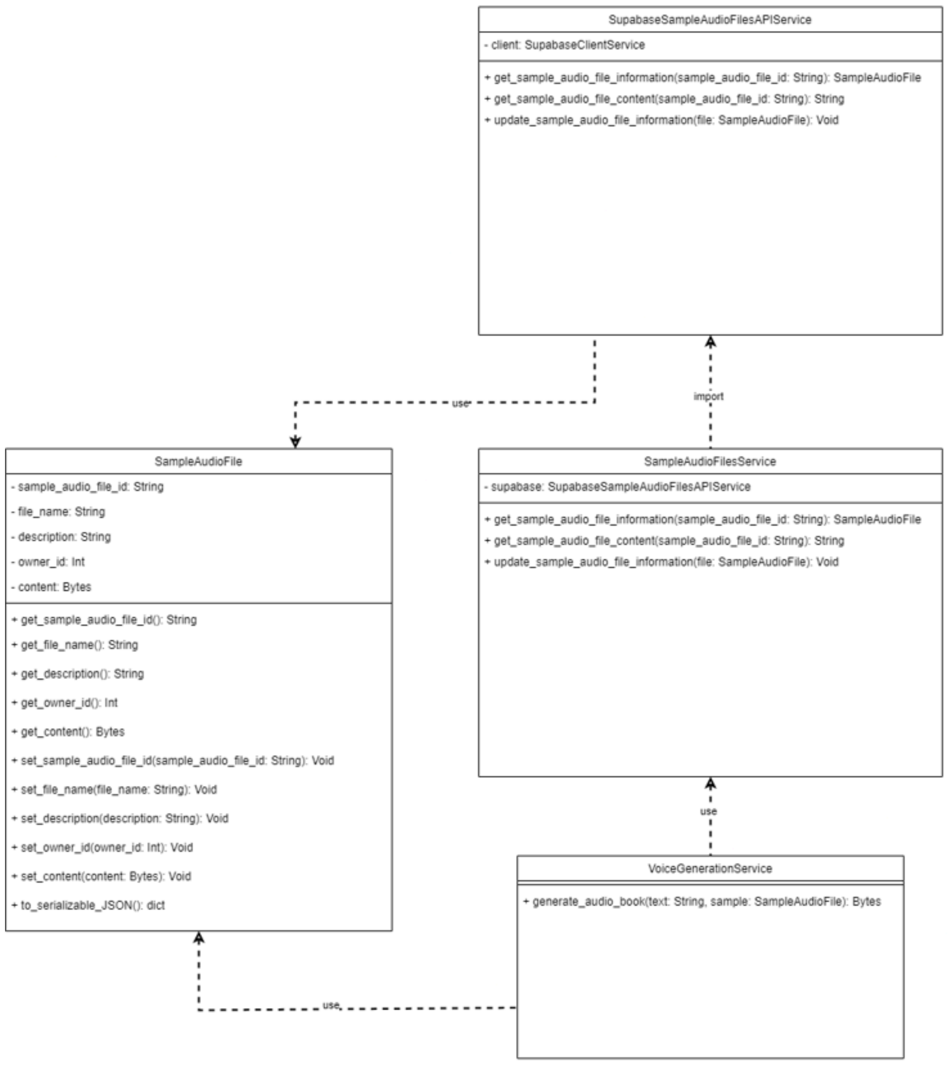
BookService: This class provides various functionalities related to managing book information, serving as the primary interface for book-related operations.

- **get_book_information:** Returns the information of a book based on its ID.
- **get_full_book_content:** Returns the full content of a book based on its ID.
- **update_book_information:** Updates the information of a book.
- **get_content_block:** Returns a specific content block of a book.

SupabaseBookAPIService: This class is responsible for implementing methods that enable the BookService to interact with the Supabase database seamlessly. It acts as a bridge between the "BookService" and the underlying Supabase database.

- **get_book_information:** Returns the information of a book from the database.
- **get_full_book_content:** Returns the full content of a book from the database.
- **update_book_information:** Updates the information of a book in the database.

4.2.2.4 Sample Audio File Services


SupabaseSampleAudioFilesAPIService
- client: SupabaseClientService
+ get_sample_audio_file_information(sample_audio_file_id: String): SampleAudioFile
+ get_sample_audio_file_content(sample_audio_file_id: String): String
+ update_sample_audio_file_information(file: SampleAudioFile): Void
use
import
SampleAudioFile
- sample_audio_file_id: String
- file_name: String
- description: String
- owner_id: Int
- content: Bytes
+ get_sample_audio_file_id(): String
+ get_file_name(): String
+ get_description(): String
+ get_owner_id(): Int
+ get_content(): Bytes
+ set_sample_audio_file_id(sample_audio_file_id: String): Void
+ set_file_name(file_name: String): Void
+ set_description(description: String): Void
+ set_owner_id(owner_id: Int): Void
+ set_content(content: Bytes): Void
+ to_serializable_JSON(): dict
SampleAudioFilesService
- supabase: SupabaseSampleAudioFilesAPIService
+ get_sample_audio_file_information(sample_audio_file_id: String): SampleAudioFile
+ get_sample_audio_file_content(sample_audio_file_id: String): String
+ update_sample_audio_file_information(file: SampleAudioFile): Void
use
VoiceGenerationService
+ generate_audio_book(text: String, sample: SampleAudioFile): Bytes
use

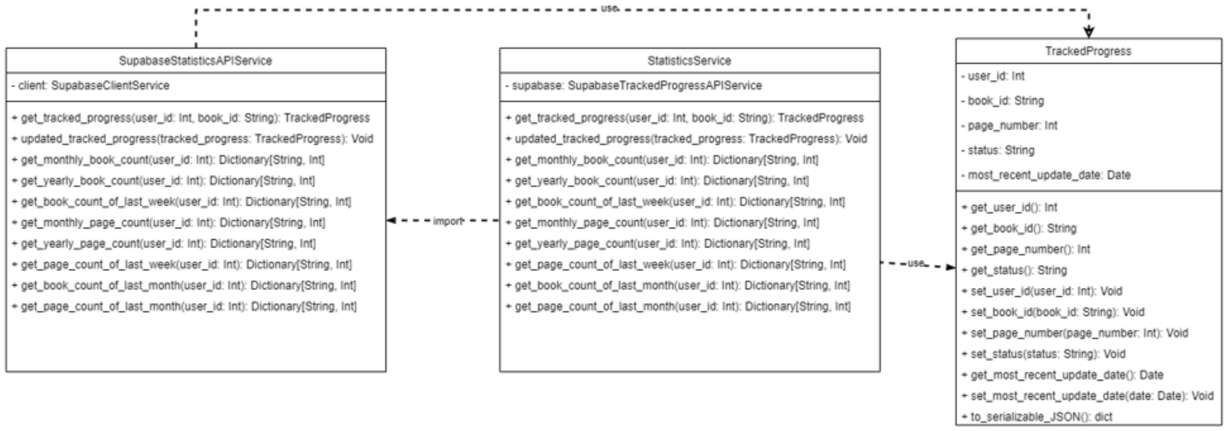
SampleAudioFilesService: This class provides various functionalities related to managing sample audio file information, serving as the primary interface for sample audio file-related operations.

- **get_sample_audio_file_information:** Returns the information of a sample audio file based on its ID.
- **get_sample_audio_file_content:** Returns the content of a sample audio file based on its ID.
- **update_sample_audio_file_information:** Updates the information of a sample audio file.

SupabaseSampleAudioFilesAPIService: This class is responsible for implementing methods that enable the SampleAudioFilesService to interact with the Supabase database seamlessly. It acts as a bridge between the "SampleAudioFilesService" and the underlying Supabase database.

- **get_sample_audio_file_information:** Returns the information of a sample audio file from the database.
- **get_sample_audio_file_content:** Returns the content of a sample audio file from the database.
- **update_sample_audio_file_information:** Updates the information of a sample audio file in the database.

4.2.2.5 Statistics Services



use
SupabaseStatisticsAPIService
- client: SupabaseClientService
+ get_tracked_progress(user_id: Int, book_id: String): TrackedProgress
+ updated_tracked_progress(tracked_progress: TrackedProgress): Void
+ get_monthly_book_count(user_id: Int): Dictionary[String, Int]
+ get_yearly_book_count(user_id: Int): Dictionary[String, Int]
+ get_book_count_of_last_week(user_id: Int): Dictionary[String, Int]
+ get_monthly_page_count(user_id: Int): Dictionary[String, Int]
+ get_yearly_page_count(user_id: Int): Dictionary[String, Int]
+ get_page_count_of_last_week(user_id: Int): Dictionary[String, Int]
+ get_book_count_of_last_month(user_id: Int): Dictionary[String, Int]
+ get_page_count_of_last_month(user_id: Int): Dictionary[String, Int]
import
StatisticsService
- supabase: SupabaseTrackedProgressAPIService
+ get_tracked_progress(user_id: Int, book_id: String): TrackedProgress
+ updated_tracked_progress(tracked_progress: TrackedProgress): Void
+ get_monthly_book_count(user_id: Int): Dictionary[String, Int]
+ get_yearly_book_count(user_id: Int): Dictionary[String, Int]
+ get_book_count_of_last_week(user_id: Int): Dictionary[String, Int]
+ get_monthly_page_count(user_id: Int): Dictionary[String, Int]
+ get_yearly_page_count(user_id: Int): Dictionary[String, Int]
+ get_page_count_of_last_week(user_id: Int): Dictionary[String, Int]
+ get_book_count_of_last_month(user_id: Int): Dictionary[String, Int]
+ get_page_count_of_last_month(user_id: Int): Dictionary[String, Int]
use
TrackedProgress
- user_id: Int
- book_id: String
- page_number: Int
- status: String
- most_recent_update_date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_page_number(): Int
+ get_status(): String
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_page_number(page_number: Int): Void
+ set_status(status: String): Void
+ get_most_recent_update_date(): Date
+ set_most_recent_update_date(date: Date): Void
+ to_serializable_JSON(): dict

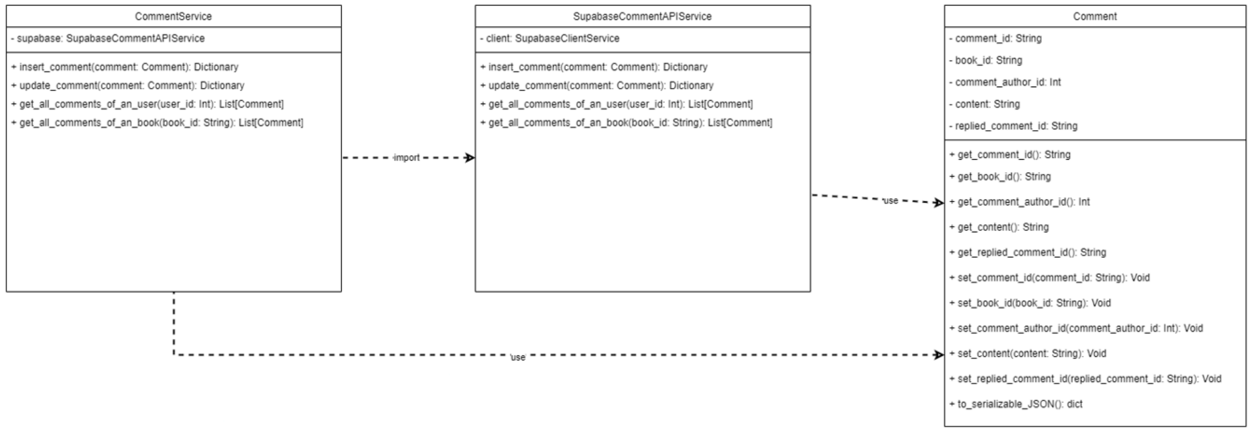
StatisticsService: This service provides various functionalities for tracking and analyzing user reading statistics.

- **get_tracked_progress:** Retrieves the reading progress of a specific book for a given user.
- **update_tracked_progress:** Updates the reading progress of a specific book for a given user.
- **get_monthly_book_count:** Calculates the number of books a user has read in a specific month.
- **get_yearly_book_count:** Calculates the number of books a user has read in a specific year.
- **get_book_count_of_last_week:** Calculates the number of books a user has read in the past week.
- **get_monthly_page_count:** Calculates the number of pages a user has read in a specific month.
- **get_yearly_page_count:** Calculates the number of pages a user has read in a specific year.
- **get_page_count_of_last_week:** Calculates the number of pages a user has read in the past week.
- **get_book_count_of_last_month:** Calculates the number of books a user has read in the past month.
- **get_page_count_of_last_month:** Calculates the number of pages a user has read in the past month.

SupabaseStatisticsAPIService: This service interacts with the Supabase database to retrieve and update user reading statistics. It provides the necessary data to the StatisticsService.

- **get_tracked_progress:** Retrieves the tracked progress data from the Supabase database.
- **update_tracked_progress:** Updates the tracked progress data in the Supabase database.
- **get_monthly_book_count:** Retrieves the monthly book count data from the Supabase database.
- **get_yearly_book_count:** Retrieves the yearly book count data from the Supabase database.
- **get_book_count_of_last_week:** Retrieves the last week's book count data from the Supabase database.
- **get_monthly_page_count:** Retrieves the monthly page count data from the Supabase database.
- **get_yearly_page_count:** Retrieves the yearly page count data from the Supabase database.
- **get_page_count_of_last_week:** Retrieves the last week's page count data from the Supabase database.
- **get_book_count_of_last_month:** Retrieves the last month's book count data from the Supabase database.
- **get_page_count_of_last_month:** Retrieves the last month's page count data from the Supabase database.

4.2.2.6 Comment Services



CommentService
- supabase: SupabaseCommentAPIService
+ insert_comment(comment: Comment): Dictionary
+ update_comment(comment: Comment): Dictionary
+ get_all_comments_of_an_user(user_id: Int): List[Comment]
+ get_all_comments_of_an_book(book_id: String): List[Comment]
import
SupabaseCommentAPIService
- client: SupabaseClientService
+ insert_comment(comment: Comment): Dictionary
+ update_comment(comment: Comment): Dictionary
+ get_all_comments_of_an_user(user_id: Int): List[Comment]
+ get_all_comments_of_an_book(book_id: String): List[Comment]
use
Comment
- comment_id: String
- book_id: String
- comment_author_id: Int
- content: String
- replied_comment_id: String
+ get_comment_id(): String
+ get_book_id(): String
+ get_comment_author_id(): Int
+ get_content(): String
+ get_replied_comment_id(): String
+ set_comment_id(comment_id: String): Void
+ set_book_id(book_id: String): Void
+ set_comment_author_id(comment_author_id: Int): Void
+ set_content(content: String): Void
+ set_replied_comment_id(replied_comment_id: String): Void
+ to_serializable_JSON(): dict
use

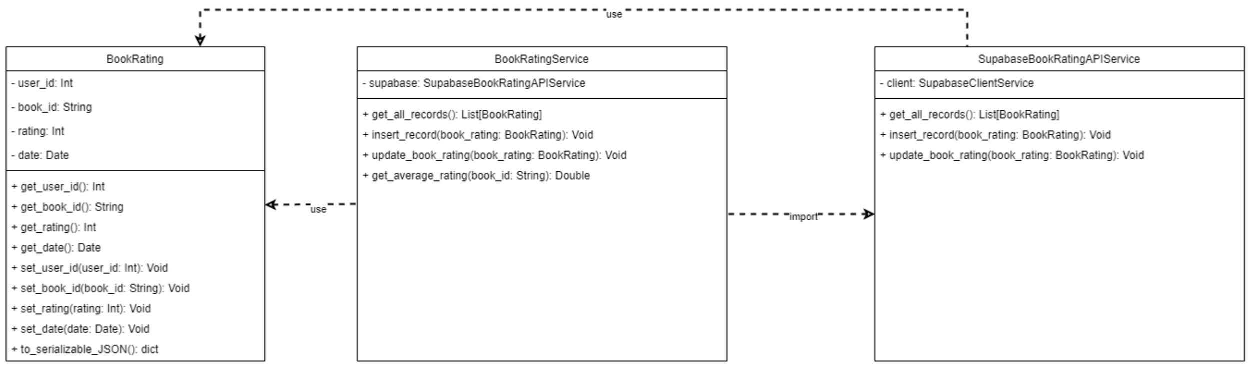
CommentService: This service provides functionalities for managing comments, including creating, updating, and retrieving comments associated with users and books.

- **insert_comment**: Inserts a new comment.
- **update_comment:** Updates an existing comment.
- **get_all_comments_of_an_user**: Retrieves all comments made by a specific user.
- **get_all_comments_of_an_book**: Retrieves all comments associated with a specific book.

SupabaseCommentAPIService: This service interacts with the Supabase database to perform comment-related operations, including creating, updating, and retrieving comments.

- **insert_comment**: Inserts a new comment into the Supabase database.
- **update_comment**: Updates an existing comment in the Supabase database.
- **get_all_comments_of_an_user**: Retrieves all comments made by a specific user from the Supabase database.
- **get_all_comments_of_an_book**: Retrieves all comments associated with a specific book from the Supabase database.

4.2.2.6 Book Rating Services



use
BookRating
- user_id: Int
- book_id: String
- rating: Int
- date: Date
+ get_user_id(): Int
+ get_book_id(): String
+ get_rating(): Int
+ get_date(): Date
+ set_user_id(user_id: Int): Void
+ set_book_id(book_id: String): Void
+ set_rating(rating: Int): Void
+ set_date(date: Date): Void
+ to_serializable_JSON(): dict
use
BookRatingService
- supabase: SupabaseBookRatingAPIService
+ get_all_records(): List[BookRating]
+ insert_record(book_rating: BookRating): Void
+ update_book_rating(book_rating: BookRating): Void
+ get_average_rating(book_id: String): Double
import
SupabaseBookRatingAPIService
- client: SupabaseClientService
+ get_all_records(): List[BookRating]
+ insert_record(book_rating: BookRating): Void
+ update_book_rating(book_rating: BookRating): Void

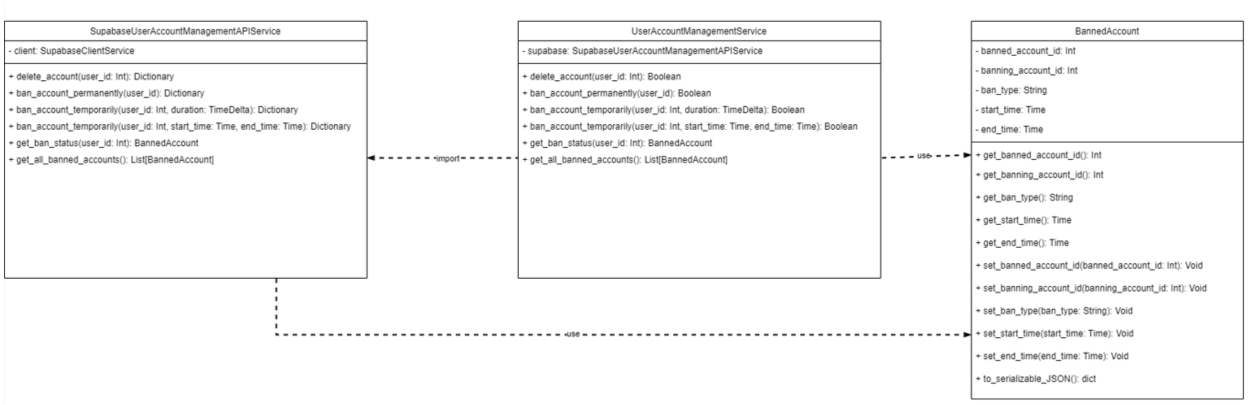
BookRatingService: This service provides functionalities for managing book ratings, including retrieving, creating, and updating ratings, as well as calculating average ratings.

- **get_all_records:** Retrieves all book ratings.
- **insert_record:** Inserts a new book rating.
- **update_book_rating:** Updates an existing book rating.
- **get_average_rating:** Calculates the average rating for a specific book.

SupabaseBookRatingAPIService: This service interacts with the Supabase database to perform book rating-related operations, including retrieving, creating, and updating ratings.

- **get_all_records:** Retrieves all book ratings from the Supabase database.
- **insert_record:** Inserts a new book rating into the Supabase database.
- **update_book_rating:** Updates an existing book rating in the Supabase database.

4.2.2.7 User Account Management Services


SupabaseUserAccountManagementAPIService
- client: SupabaseClientService
+ delete_account(user_id: Int): Dictionary
+ ban_account_permanently(user_id): Dictionary
+ ban_account_temporarily(user_id: Int, duration: TimeDelta): Dictionary
+ ban_account_temporarily(user_id: Int, start_time: Time, end_time: Time): Dictionary
+ get_ban_status(user_id: Int): BannedAccount
+ get_all_banned_accounts(): List[BannedAccount]
import
UserAccountManagementService
- supabase: SupabaseUserAccountManagementAPIService
+ delete_account(user_id: Int): Boolean
+ ban_account_permanently(user_id): Boolean
+ ban_account_temporarily(user_id: Int, duration: TimeDelta): Boolean
+ ban_account_temporarily(user_id: Int, start_time: Time, end_time: Time): Boolean
+ get_ban_status(user_id: Int): BannedAccount
+ get_all_banned_accounts(): List[BannedAccount]
use
BannedAccount
- banned_account_id: Int
- banning_account_id: Int
- ban_type: String
- start_time: Time
- end_time: Time
+ get_banned_account_id(): Int
+ get_banning_account_id(): Int
+ get_ban_type(): String
+ get_start_time(): Time
+ get_end_time(): Time
+ set_banned_account_id(banned_account_id: Int): Void
+ set_banning_account_id(banning_account_id: Int): Void
+ set_ban_type(ban_type: String): Void
+ set_start_time(start_time: Time): Void
+ set_end_time(end_time: Time): Void
+ to_serializable_JSON(): dict
use

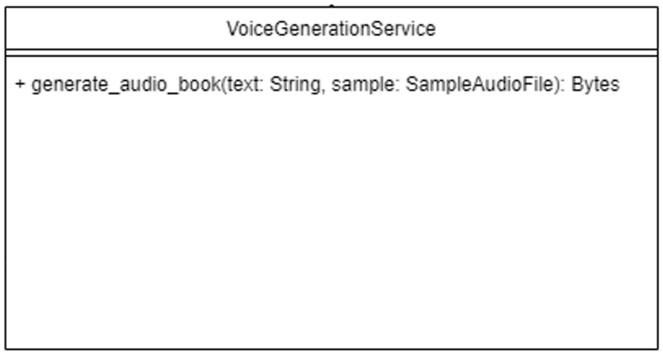
UserAccountManagementService: This service provides functionalities to manage permissions of user accounts.

- **delete_account:** Deletes a user account.
- **ban_account_permanently:** Permanently bans a user account.
- **ban_account_temporarily:** Temporarily bans a user account for a specified duration.
- **get_ban_status:** Retrieves the ban status of a user account.
- **get_all_banned_accounts:** Retrieves a list of all banned accounts.

SupabaseUserAccountManagementAPIService: This service interacts with the Supabase database to perform user account permission management operations.

- **delete_account:** Deletes a user account from the Supabase database.
- **ban_account_permanently:** Permanently bans a user account in the Supabase database.
- **ban_account_temporarily:** Temporarily bans a user account in the Supabase database.
- **get_ban_status:** Retrieves the ban status of a user account from the Supabase database.
- **get_all_banned_accounts:** Retrieves a list of all banned accounts from the Supabase database.

4.2.2.8 Voice Generation Services



VoiceGenerationService
+ generate_audio_book(text: String, sample: SampleAudioFile): Bytes

VoiceGenerationService: This service provides a function to generate audio for a given text input using a specified sample audio file.

- **generate_audio_book:** Generates audio of voice reading a specific book from text input using a sample audio file.

4.2.3 Routes

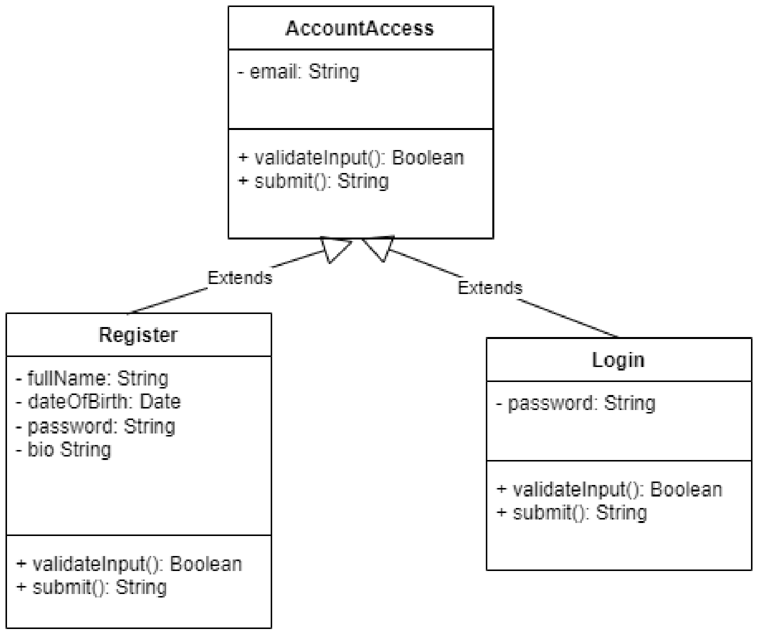
This sub-component is responsible for defining the API endpoints that expose the application's functionality to external clients. It serves as the bridge between the client-facing presentation layer and the backend logic of the application. Routes are designed to handle incoming requests, delegate tasks to appropriate services, and deliver well-structured responses to clients, ensuring seamless communication and interaction.

The implementation of this sub-component consists of multiple files, each dedicated to defining specific routes and their associated logic.

4.3 Component: Presentation Tier

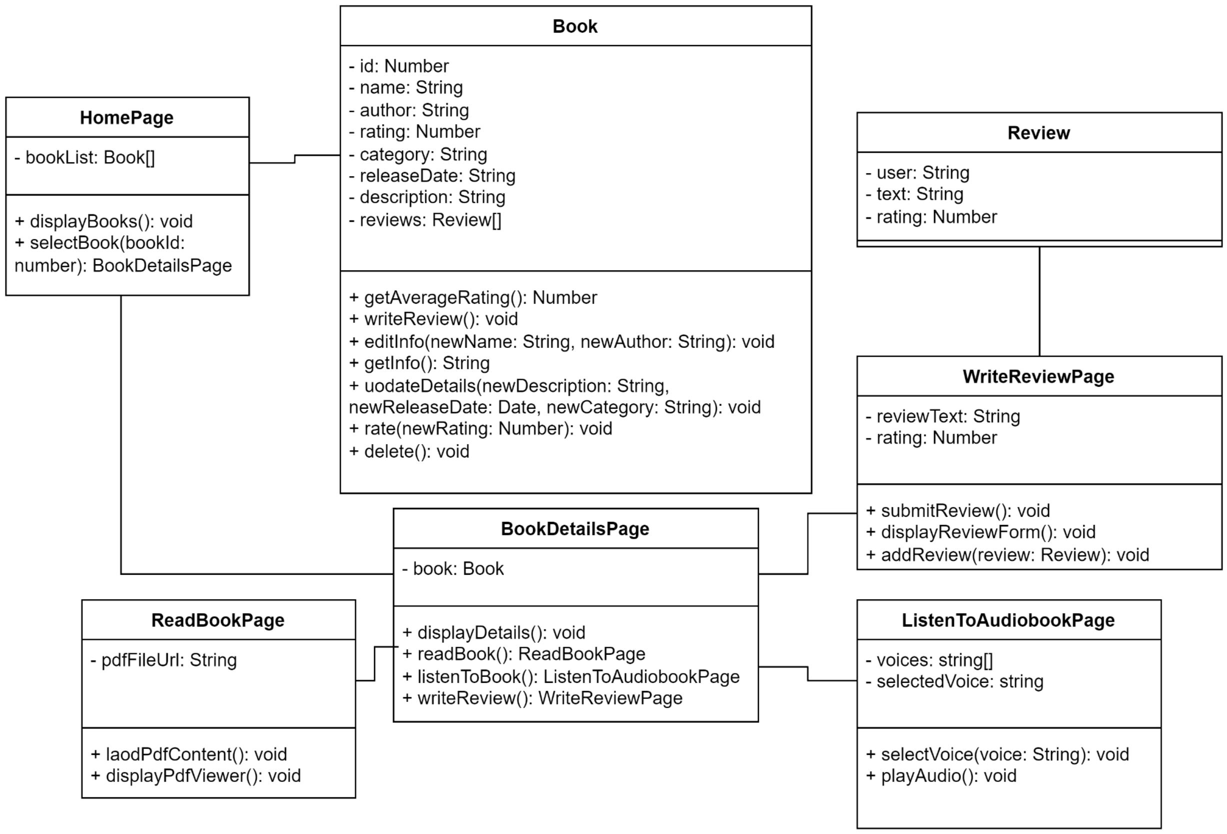
The Typescript programming language and the React library is used to implement this sub-component.

4.3.1 Login/Register Interface Management



AccountAccess: provide the common property and shared method for dealing with user input and validation.
Register: Implement methods to ensure all fields are correctly filled and meet the requirements, then send data to the Application Logic Tier for processing.
Login: Implement methods to check the login data are correct and send data to the Application Logic Tier for authentication.

4.3.2 Book Reading and Listening Interface Management

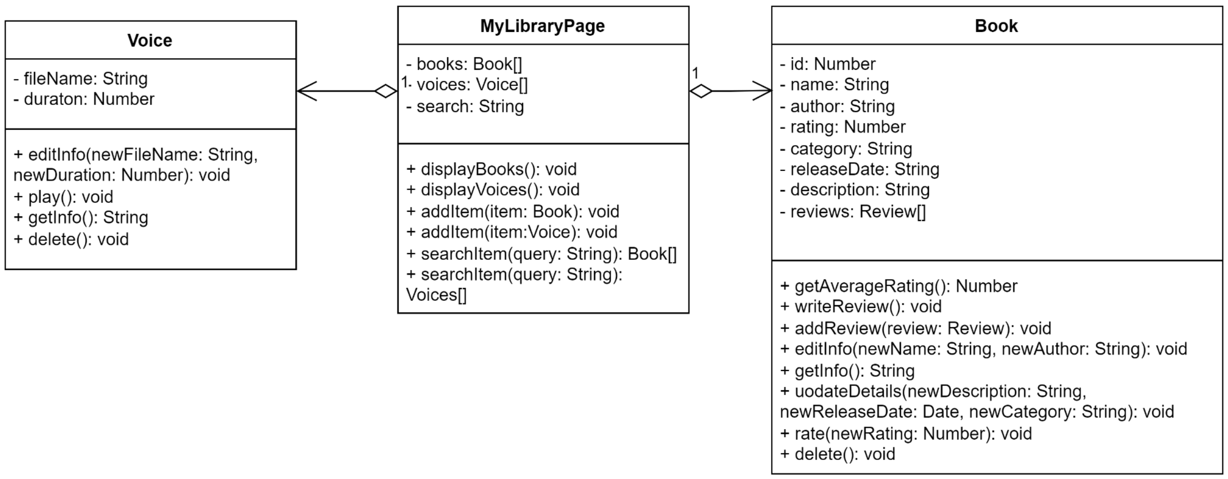


HomePage: Provides methods for rendering the list of books and selecting a book on the home page.
Book: Calculates the average rating from the review array and adds a new review to the review array.
BookDetailsPage: Displays detailed information about a selected book, including the option to:

- Read the book
- Listen to the audiobook
- Write a review

ReadBookPage: Displays the text content of the selected book for reading.
ListenToAudiobookPage: Provides audiobook playback functionality and allows users to select a narration voice.
WriteReviewPage: Provides a text field for the user to write a review and a rating bar for feedback, then sends the user input to the Application Logic Tier for processing.
Review: Represents a review written by a user, including user, text, rating and acts as a component of the Book class, providing feedback data.

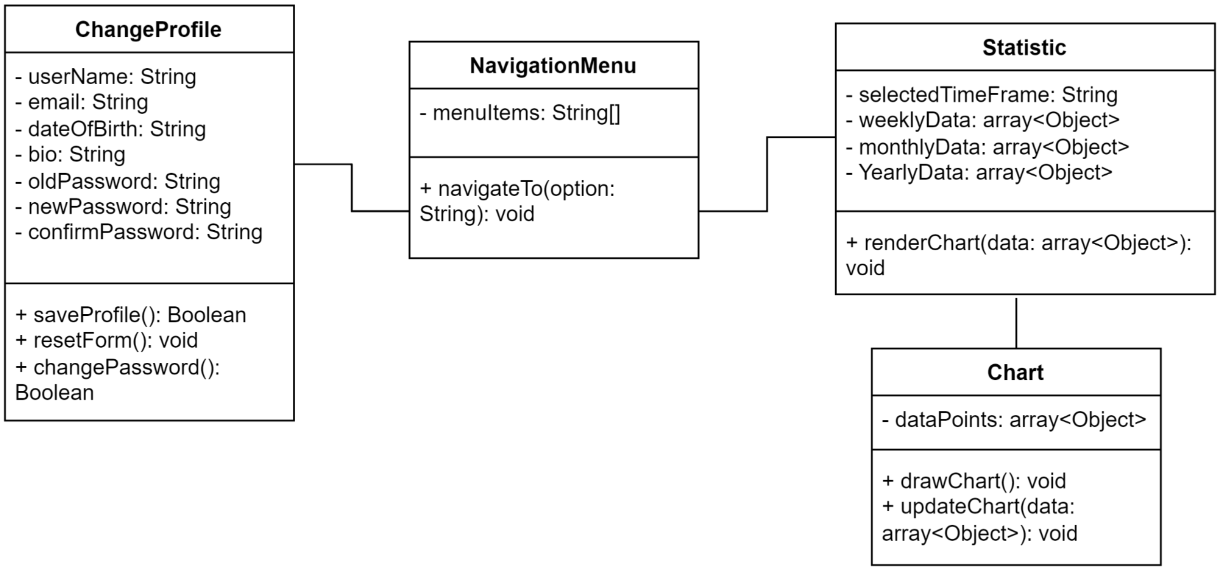
4.3.3 My Library Interface Management



Voice
- fileName: String
- duraton: Number
+ editInfo(newFileName: String, newDuration: Number): void
+ play(): void
+ getInfo(): String
+ delete(): void
MyLibraryPage
- books: Book[]
- voices: Voice[]
- search: String
+ displayBooks(): void
+ displayVoices(): void
+ addItem(item: Book): void
+ addItem(item:Voice): void
+ searchItem(query: String): Book[]
+ searchItem(query: String): Voices[]
1
1
Book
- id: Number
- name: String
- author: String
- rating: Number
- category: String
- releaseDate: String
- description: String
- reviews: Review[]
+ getAverageRating(): Number
+ writeReview(): void
+ addReview(review: Review): void
+ editInfo(newName: String, newAuthor: String): void
+ getInfo(): String
+ uodateDetails(newDescription: String, newReleaseDate: Date, newCategory: String): void
+ rate(newRating: Number): void
+ delete(): void

MyLibraryPage: Represents the "My Library" page where users manage their collections of books and voice files.
Book: Represents a book object that holds all details related to a book across different pages of the website.
Voice: Represents a voice file added by the user for audiobook purposes.

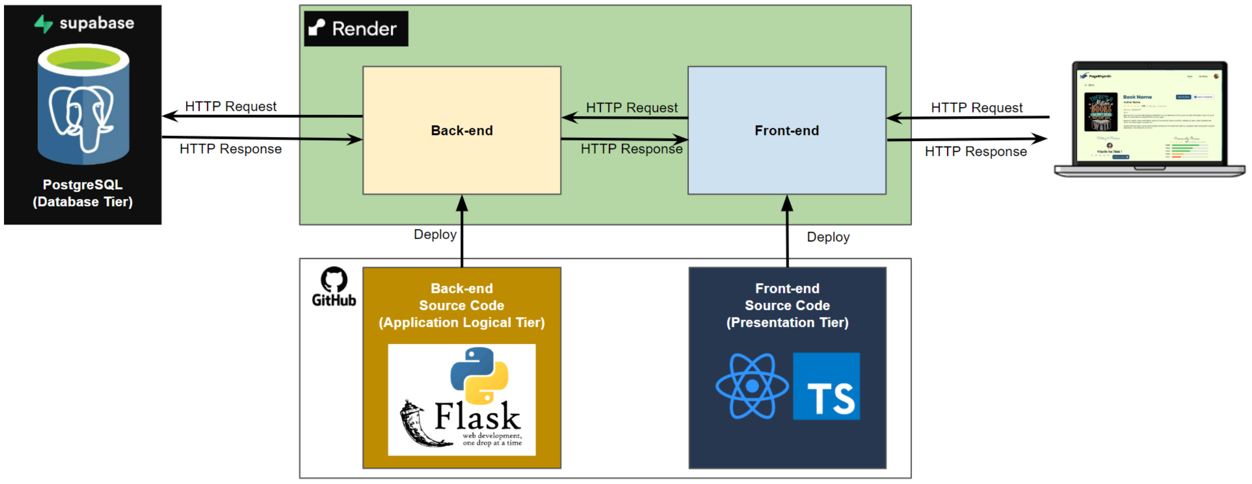
4.3.4 Profile Interface Management



ChangeProfile
- userName: String
- email: String
- dateOfBirth: String
- bio: String
- oldPassword: String
- newPassword: String
- confirmPassword: String
+ saveProfile(): Boolean
+ resetForm(): void
+ changePassword(): Boolean
NavigationMenu
- menuItems: String[]
+ navigateTo(option: String): void
Statistic
- selectedTimeFrame: String
- weeklyData: array<Object>
- monthlyData: array<Object>
- YearlyData: array<Object>
+ renderChart(data: array<Object>): void
Chart
- dataPoints: array<Object>
+ drawChart(): void
+ updateChart(data: array<Object>): void

NavigationMenu: Handles navigation between different options on the profile page, such as "Edit Profile," "View Statistics," and "Logout".
ChangeProfile: Represents the profile-editing section of the page where the user can update personal details like name, email, date of birth, bio and change password.
Statistic: Manage the functionality of the “View Statistic” part, which displays the user’s weekly, monthly and yearly statistics.
Chart: Handles the graphical representation of the statistics data.

5. Deployment


supabase
PostgreSQL (Database Tier)
HTTP Request
HTTP Response
Render
Back-end
HTTP Request
HTTP Response
Front-end
HTTP Request
HTTP Response
Deploy
Deploy
GitHub
Back-end Source Code (Application Logical Tier)
Flask
web development, one drop at a time
Front-end Source Code (Presentation Tier)
TS

The diagram above illustrates the system's deployment architecture (links to the deployment platforms are provided in the reference subsection of the first section).

- The database, implemented using PostgreSQL, is hosted on Supabase.
- The application logic tier's source code is stored in GitHub and deployed on Render for execution.
- Similarly, the presentation tier's source code is also managed in GitHub and deployed on Render, ensuring seamless integration and operation across all layers.

In summary, the architecture comprises four main nodes that interact with each other via the HTTP protocol:

- **Database Node**: Hosted on web servers provided by Supabase, it manages the system's data storage and retrieval.
- **Back-End Node**: Deployed on web servers managed by Render, this node handles the core application logic.
- **Front-End Node**: Also deployed on web servers managed by Render, this node manages the presentation layer, serving user interface files and content.
- **Desktop-Based Web Clients**: These clients request resources from the Front-End Node, which responds with rendered files that are displayed to the users.

This architecture ensures efficient communication and a cohesive workflow across all layers of the system.

6. Implementation View

The project folder structure is organized as follows, with “src” serving as the root folder:

```
src/
├── BackEnd/
│   └── PageRhythm/
│       ├── app.py
│       ├── routes/
│       │   ├── __init__.py
│       │   ├── home_route.py
│       │   ├── book_route.py
│       │   ├── sample_audio_file_route.py
│       │   ├── statistics_route.py
│         │    ├── comment_route.py
│         │    ├── book_rating_route.py
│         │    ├── tracked_progress_route.py
│         │    ├── user_account_management_route.py
│         │    └── voice_generation_route.py
│         ├── models/
│         │    ├── __init__.py
│         │    ├── base_entity.py
│         │    ├── account.py
│         │    ├── book.py
│         │    ├── comment.py
│         │    ├── sample_audio_file.py
│         │    ├── book_rating.py
│         │    ├── tracked_progress.py
│         │    └── banned_account.py
│         └── services/
│              ├── __init__.py
│              ├── supabase_client_service.py
│              ├── account/
│              │    ├── account_service.py
│              │    └── supabase_account_api_service.py
│              ├── authentication/
│              │    ├── authentication_service.py
│              │    └── supabase_authentication_api_service.py
│              ├── book/
│              │    ├── book_service.py
│              │    └── supabase_book_api_service.py
│              ├── sample_audio_file/
│              │    ├── sample_audio_file_service.py
│              │    └── supabase_sample_audio_file_api_service.py
│              ├── statistics/
│              │    ├── statistics_service.py
│              │    └── supabase_statistics_api_service.py
│              ├── comment/
│              │    ├── comment_service.py
│              │    └── supabase_comment_api_service.py
│              ├── book_rating/
│              │    ├── book_rating_service.py
│              │    └── supabase_book_rating_api_service.py
│              ├── user_account_management/
│              │    ├── user_account_management_service.py
│              │    └── supabase_user_account_management_api_service.py
│              └── voice_generation/
│                   └── voice_generation_service.py
└── FrontEnd/
     └── PageRhythm/
          ├── index.html
          ├── global.d.ts
          └── src/
               ├── main.tsx
               ├── App.tsx
               ├── Images.ts
               ├── Server.ts
               ├── index.css
               │
```

```
│── App.css
│── components/
│   │── BookDetailsPage.tsx
│   │── BooksMyLibraryPage.tsx
│   │── CommentPage.tsx
│   │── GeneralProfilePage.tsx
│   │── HomePage.tsx
│   │── LandingPage.tsx
│   │── ListenToBookPage.tsx
│   │── MyLibrarySectionBar.tsx
│   │── NavigationBar.tsx
│   │── PasswordProfilePage.tsx
│   │── ProfileSectionBar.tsx
│   │── ReadBookPage.tsx
│   │── RegisterPage.tsx
│   │── RequestPasswordResetPage.tsx
│   │── StatisticsProfilePage.tsx
│   └── VoicesMyLibraryPage.tsx
└── styles/
    │── book-details-page-styles.css
    │── books-my-library-page-styles.css
    │── comment-page-styles.css
    │── fonts.css
    │── general-profile-page-styles.css
    │── home-page-styles.css
    │── landing-page-styles.css
    │── listen-to-book-page-styles.css
    │── my-library-section-bar-styles.css
    │── navigation-bar-styles.css
    │── password-profile-page-styles.css
    │── profile-section-bar.css
    │── read-book-page-styles.css
    │── register-page-styles.css
    │── request-password-reset-page-styles.css
    │── statistics-profile-page-styles.css
    └── voices-my-library-page-styles.css
```