Bilkent University
Department of Computer Engineering

CS 491- Senior Design Project

Papyrus

Project Name: Papyrus

Low-Level Design Report

Group Members: Ant Duru, Atay Kaylar, Enver Yiğitler
Supervisor: Fazlı Can

Jury Members: Shervin Arashloo, Hamdi Dibeklioğlu

Innovation Expert: Murat Kalender

1. Introduction

Learning or mastering a new language is an important instrument for enhancing one's personal growth and has a number of pragmatic uses in intrinsic and extrinsic domains. It improves job opportunities, sets the stage for obtaining more complex relationships with foreign people or even providing a deeper understanding of self in a person[1]. One of the greatest authors Johann Wolfgang von Goethe would say "He who knows no foreign languages knows nothing of his own.", where studying language opens the door for deeper understanding of various cultures through its embeddedness into a nation's collective unconscious. With understanding of others, one can comprehend itself more profoundly. Hence we as a team knew the importance of language learning and opted to create an application that helps people to further train on a specific language. *Duolingo* is the most popular language learning application in the market[2], it helps people to learn new languages with a variety of game-like exercises. When it comes to learning words or fundamental structures, it is very useful, however it is not enough for mastery[3]. Kató Lomb is a Hungarian multilingual that has self-taught herself 16 languages. In order for her to master or even start to learn new languages, she insisted on learning the context of sentences, thereby the complex expressions, or the unknown words were redundant for her[4]. Papyrus provides a word learning tool, where the user defines flashcards, however we acknowledge the fact that learning a language comes from reading and understanding the context just like Lomb addressed, and we constructed our application according to it. Hence creating a more effective program that lets its users master languages, better than any other application on the market.

1.1 Object design trade-offs

Functionality vs. Usability

As explained in the introduction section, we aim to make an application that users can master or progress in their preferred language in the most effective way. Although our application has a number of language learning tools, hence the number of functionalities. User experience is a top priority to us. In order to make the application more usable and understandable, we might sacrifice some of the functionality during the development or in the future.

Cost vs. Robustness

Papyrus is a senior year graduation project that is being made by three students. Thereby, there are money, time and labor-power limitations. We opt to make the best we can do, however limitations can decrease the robustness of the project.

Rapid Development vs. Functionality

We aim to make the program with the intended functionalities, without considering making the project a rapid development environment. We said that usability is more important than functionality, however the program will not be complete without the intended functionality. Thus, it is one of our top priorities.

Reliability vs. Robustness

One of the most prominent features of the project is the sentence alignment between different translations of the selected books. Bulk of the learning will occur during inspecting the aligned sentences, therefore the correctness of the alignment is crucial for program's effectiveness. In that we aim to make the alignment operation as accurate as we can, with minimal time constraints in the hardware and the software, hindering the robustness of the sentence alignment operations.

Portability vs. Low Development Time

We aim to make Papyrus an iOS application, where the iPhone implementation is our main concern. Thanks to Swift, implementing the program across different Apple platforms is less cumbersome than other programming languages(e.g. Java, C++). However due to the limitations of the development process, we will test and implement the system mainly on the iPhone platform. However, portability limitations will have a tendency to be improved after the CS492, with further implementation on desktop systems, or maybe on Android systems, with a whole new programming language.

1.2 Interface documentation guidelines

In this report, we will use the following table to demonstrate our classes, their properties and functions.

class className
class definition
Properties
-attribute1: attribute_type -attribute2: attribute_type
Functions
+function_name(function_param: param_type): return_type +function_name(function_param: param_type): return_type

Figure 1: Interface Documentation Guidelines

In properties and functions “-” means private, “+” means public.
“none” means the non-existence of property or function

1.3 Engineering standards (e.g., UML and IEEE)

In this report, we use IEEE reference style to cite our sources. Our diagrams are created following the UML guidelines.

1.4 Definitions, acronyms, and abbreviations

Database: A database is an organized collection of structured information, or data, typically stored electronically in a computer system. [5]

Reader: The organizer for in-book actions in the project.

Library: Holder for the books and the metadata of the books in the project.

Data: Information about every component of the project which interests the user and the client.

Flashcard: a card bearing words, numbers, or pictures that is briefly displayed (as by a teacher to a class) usually as a learning aid. [6]

Property: the characteristics of an object.

Function: a block of organized, reusable code that is used to perform a single, related action.[7]

Swift: Programming language used mainly for macOS and iOS platforms. Papyrus is written in Swift.

Android: A phone operating system.

2. Packages

2.1 Data

This subsystem manages the local subsystem, in that way it manages the operations within the program's properties (Eg. User data, book data, shelf data…).

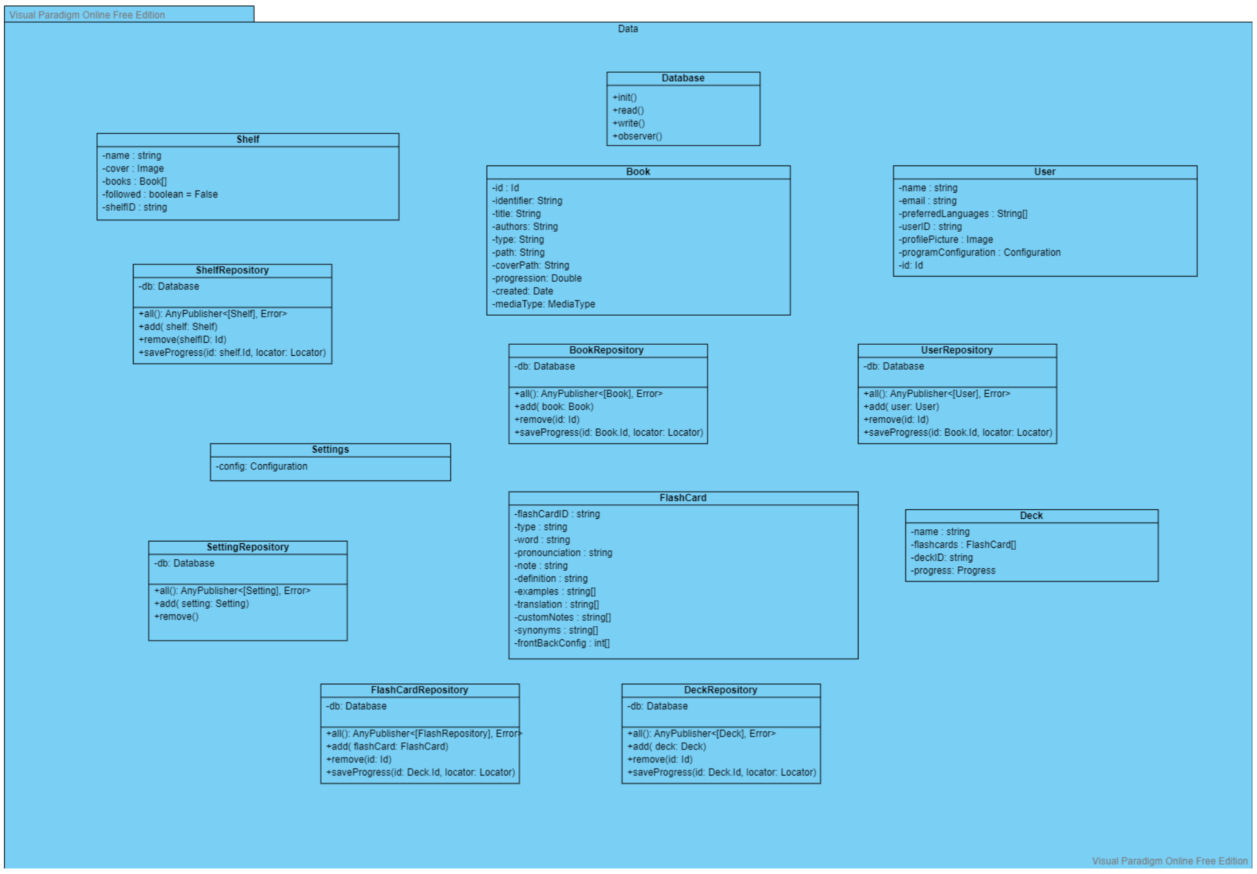

Figure 2: Data Package

Shelf

This class is responsible for holding the user’s books, which is one of his/her personal bookshelves. It holds the information about the books.

Database

This package is the client database of the application. Therefore we need an intermediary class to communicate with the database. This class is simply named database, which has few functions to communicate with databases.

User

This class is responsible for holding the users’ information.

Book

This class is responsible for holding individual book’s necessary information.

ShelfRepository

This class wraps the database and its functions, so that the program can use the database operations on the shelf objects.

BookRepository

This class wraps the database and its functions, so that the program can use the database operations on the book objects.

UserRepository

This class wraps the database and its functions, so that the program can use the database operations on the user objects.

Settings

This class holds the program settings information via the configuration class.

Flashcard

The class is responsible for holding the information of the flashcard objects, holding its desired data members.

Deck

This class holds the information of the deck, which holds the number of flashcards inside it.

SettingsRepository

This class wraps the database and its functions, so that the program can use the database operations on the shelf objects.

FlashCardRepository

This class wraps the database and its functions, so that the program can use the database operations on the flashcard objects.

DeckRepository

This class wraps the database and its functions, so that the program can use the database operations on the deck objects.

2.2 Library

This package manages the user's books, audiobooks and shelves, by using the information from the database it visualizes the properties with using view controller.

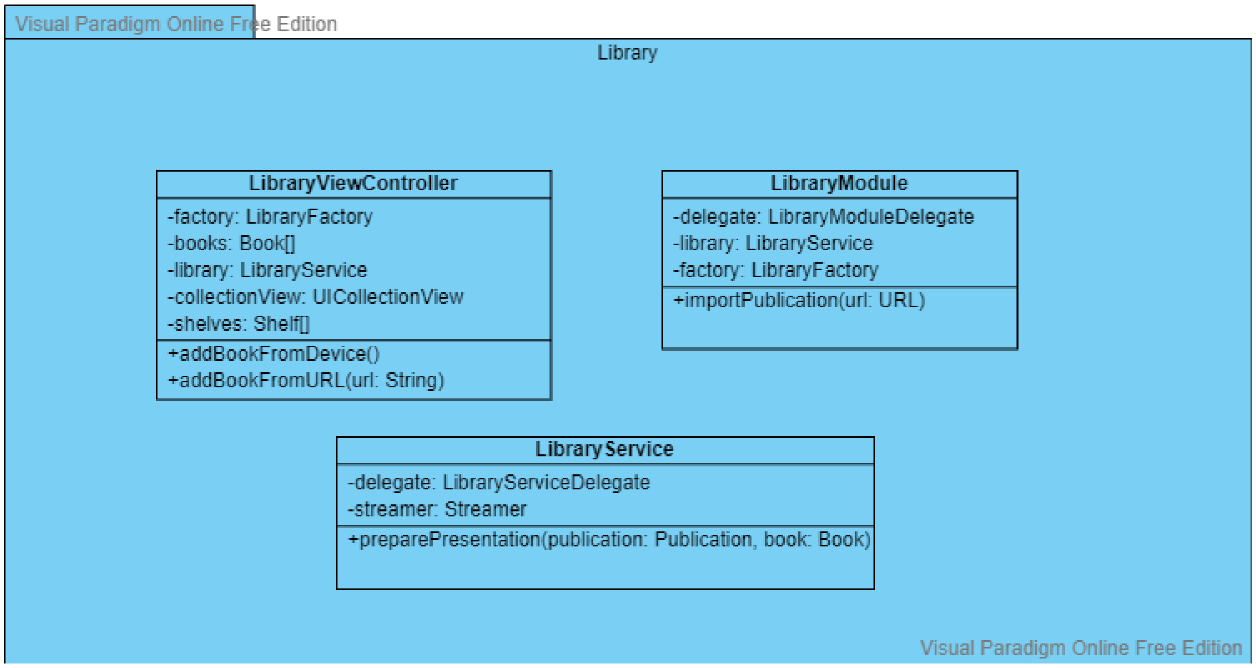
Visual Paradigm Online Free Edition
Library
LibraryViewController
-factory: LibraryFactory
-books: Book[]
-library: LibraryService
-collectionView: UICollectionView
-shelves: Shelf[]
+addBookFromDevice()
+addBookFromURL(url: String)
LibraryModule
-delegate: LibraryModuleDelegate
-library: LibraryService
-factory: LibraryFactory
+importPublication(url: URL)
LibraryService
-delegate: LibraryServiceDelegate
-streamer: Streamer
+preparePresentation(publication: Publication, book: Book)
Visual Paradigm Online Free Edition

Figure 3: Library Package

LibraryViewController

Presents the library and in the models shows the audiobooks and book using the library service

LibraryModule

It is wrapper class of the other library classes, that is used for constructing the library

LibraryService

The class is used for the logic of the library

2.3 Reader

The model and the view controller part of the program that shows the user, audiobooks, books and the narration mode, with the additional UIKit features.

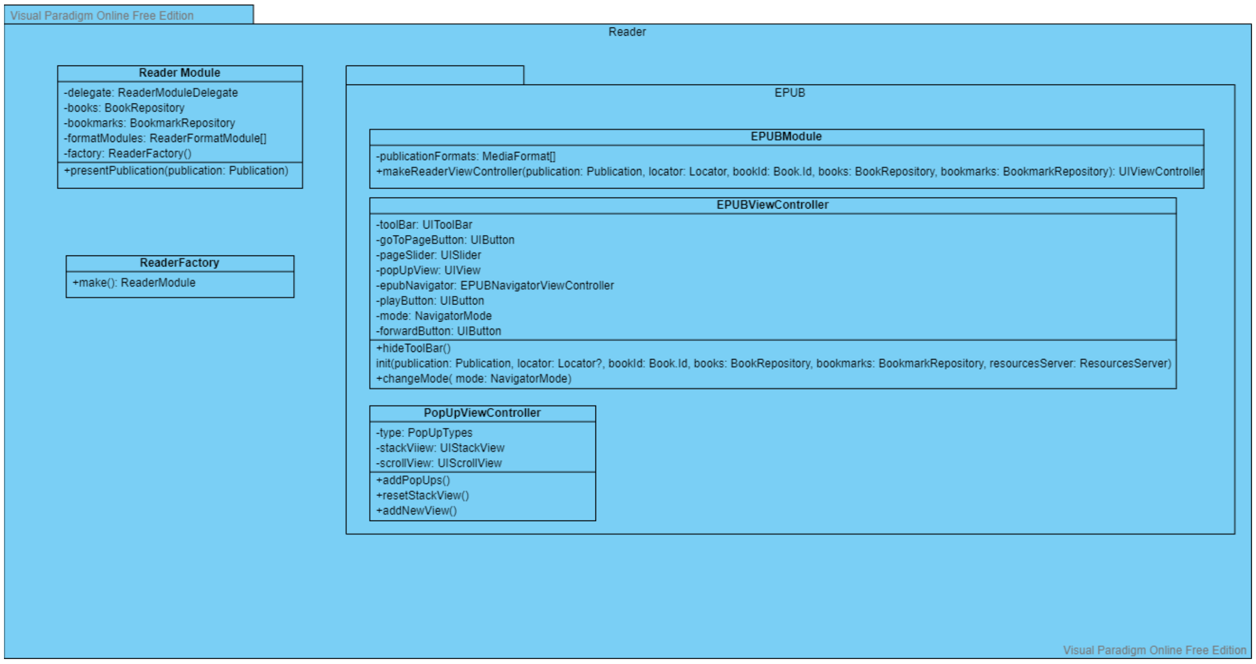
Visual Paradigm Online Free Edition
Reader
Reader Module
-delegate: ReaderModuleDelegate
-books: BookRepository
-bookmarks: BookmarkRepository
-formatModules: ReaderFormatModule[]
-factory: ReaderFactory()
+presentPublication(publication: Publication)
ReaderFactory
+make(): ReaderModule
EPUB
EPUBModule
-publicationFormats: MediaFormat[]
+makeReaderViewController(publication: Publication, locator: Locator, bookId: Book.Id, books: BookRepository, bookmarks: BookmarkRepository): UIViewController
EPUBViewController
-toolBar: UIToolBar
-goToPageButton: UIButton
-pageSlider: UISlider
-popUpView: UIView
-epubNavigator: EPUBNavigatorViewController
-playButton: UIButton
-mode: NavigatorMode
-forwardButton: UIButton
+hideToolBar()
init(publication: Publication, locator: Locator?, bookId: Book.Id, books: BookRepository, bookmarks: BookmarkRepository, resourcesServer: ResourcesServer)
+changeMode(mode: NavigatorMode)
PopUpViewController
-type: PopUpTypes
-stackViiew: UIStackView
-scrollView: UIScrollView
+addPopUps()
+resetStackView()
+addNewView()
Visual Paradigm Online Free Edition

Figure 4: Reader Package

Reader Module

The logic of the reader package

Reader Factory

Creates the reader module

EPUBModule

2.4 FlashCard

ViewController package that manages the visualizing and the logic behind flashcards and the decks, that is comprised of flashcards.

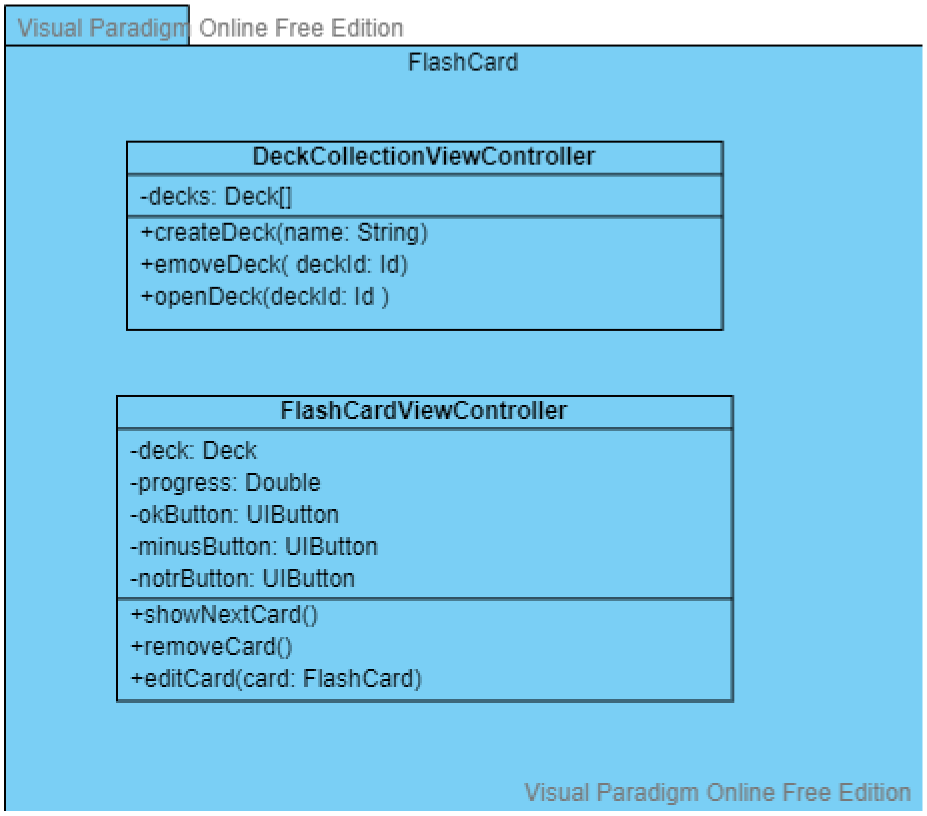

Figure 5: FlashCard Package

DeckCollectionViewController

Models and views the decks

FlashCardViewController

Models the logic behind it and views the flashcardsCreates the viewcontroller for the operations of the delegate objects.

EPUBViewController

Presents the books and the audiobook and in the model shows the audiobooks and book using the library service

PopUpViewController

Presents the popups and with a model. views the popups.

3. Class Interface

3.1 Data

Shelf

class Shelf
This class is responsible for holding the user's books, which is one of his/her personal bookshelves. It holds the information about the books.
Properties
-name: string : Name of the shelf -cover: image : Cover of the shelf, which is type image -books: Book[] : Array of type book, which includes the books in the shelf -followed: boolean = False : Boolean value, which holds whether the shelf is followed by the user or not. -shelfID: string : ID of the shelf
Functions
none

Book

class Book
This class is responsible for holding individual book’s necessary information.
Properties
-id: id : ID of the book which is type Id -identifier: String : Another identifier which is of type string -title: String : Holds the title of the book -authors: String : Holds the name(s) of the author(s) in type string -type: String : Holds the type or genre of the book as a string -path: String : Path of the book -coverPath: String: Path of the cover of the book -progression: Double: Percentage of the read part of the book -created: Date : The books creation date in database in type Date -mediaType: MediaType : Media format of the book, whether it is text or audio
Functions
none

User

Class User
This class is responsible for holding the users’ information.
Properties
-name: String : Full name of the user -email: String : Email of the user -preferredLanguages : String[] : Languages which the user chose -userID : String : ID of the user -profilePicture : Image : Profile picture of the user in Image type -programConfiguration : Configuration :Configuration of the program in configuration type -id: Id : ID of the user in Id type.
Functions
none

ShelfRepository

class ShelfRepository
This class wraps the database and its functions, so that the program can use the database operations on the shelf objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[Shelf], Error> : Object of Combine framework +add(shelf: Shelf) : It is used to add shelves into the database +remove(shelfID: Id) : It is used to remove shelves from the database

BookRepository

Class BookRepository
This class wraps the database and its functions, so that the program can use the database operations on the book objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[Book], Error> : Object of Combine framework +add(book: Book) : It is used to add books into the database +remove(id: Id) : It is used to remove books from the database +saveProgress(id: Book.Id, locator: Locator) : This function is used for saving the page progress in a book, with passing the id of the book and the locator which locates the progress.

UserRepository

Class UserRepository
This class wraps the database and its functions, so that the program can use the database operations on the user objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[User], Error> : Object of Combine framework +add(shelf: Shelf) : It is used to add shelves into the database +remove(shelfID: Id) : It is used to remove shelves from the database

FlashCard

class FlashCard
The class is responsible for holding the information of the flashcard objects, holding its desired data members.
Properties
-flashCardID : String : ID of the flashcard -type : String : type of the flashcard -word : String : Word of the flashcard -pronunciation : String : Pronunciation of the word, which writes on the flashcard -note : String : special note on the flashcard -definition : String : Definition of the word on the flashcard -examples : String[] : Array of examples -translation : String[] : Translation of the word -customNotes: String[] : Extra notes that the user can add to a flashcard -synonyms: String[]: Synonyms for flashcard’s word -frontBackConfig : int[] : Array of integers that are used to configure the flashcard property visibility(properties are adjustable).
Functions
none

Deck

class Deck
This class holds the information of the deck, which holds the number of flashcards inside it.
Properties
-name: String : Name of the deck -flashcards: FlashCard[] : Array of flashcards inside the deck -deckID : String : Holds the id of the deck as a string -progress : Progress : User's progress of the deck
Functions
none

DeckRepository

class DeckRepository
This class wraps the database and its functions, so that the program can use the database operations on the deck objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[Deck], Error> : Object of Combine framework +add(deck : Deck) : It is used to add decks into the database +remove(id: Id) : It is used to remove decks from the database +saveProgress(id: Deck.Id, locator: Locator) : This function is used for saving the deck progress , with passing the id of the deck and the locator which locates the progress. As you can see the deck is composed of flashcards.

FlashCardRepository

class FlashCardRepository
This class wraps the database and its functions, so that the program can use the database operations on the flashcard objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[FlashCard], Error> : Object of Combine framework +add(flashCard : FlashCard) : It is used to add decks into the database +remove(id: Id) : It is used to remove decks from the database

Settings

class Settings
This class holds the program settings information via the configuration class.
Properties
-config: Configuration :Configurations of the settings and the program through the Configuration class
Functions
none

SettingsRepository

class SettingsRepository
This class wraps the database and its functions, so that the program can use the database operations on the shelf objects.
Properties
-db: Database : Database object in the repository, which is used to wrap the functions of the database class
Functions
+all(): AnyPublisher<[Setting], Error> : Object of Combine framework +add(setting : Setting) : It is used to adjust a setting in the database +remove() : It is used to remove a setting in the database

Database

class Database
This package is the client database of the application. Therefore we need an intermediary class to communicate with the database. This class is simply named database, which has few functions to communicate with databases.
Properties
+init() : This function is used to initialize the database
Functions
+read() : It is used to read the database +write() : It is used to write into the database +observer(): Database observer

3.2 Library

LibraryViewController

class LibraryViewController
Presents the library and in the models shows the audiobooks and book using the library service
Properties
-factory : LibraryFactory : A factory property that is used to create library -books : Book[] : Books in the library held as an array of type Book -library : LibraryService :Library of the user of type LibraryService -collectionView : UICollectionView : Collection view that is used for presenting the UI -shelves : Shelf[] : Shelves of the user helds as an array
Functions
+addBookFromDevice() :This function adds books to library from the local db of the database +addBookFromURL(url: String) :This function adds book to library with a given URL that is passed as a parameter to function

LibraryModule

class LibraryModule
It is wrapper class of the other library classes, that is used for constructing the library
Properties
-delegate : LibraryModuleDelegate : Delegate of the library for communication between other classes -library: LibraryService : Library of the user of type LibraryService -factory: LibraryFactory: A factory property that is used to create library
Functions
+importPublication(url : URL): It is used to import a publication to library with a given URL

LibraryService

class LibraryService
The class is used for the logic of the library
Properties
-delegate : LibraryServiceDelegate : Delegate of the LibraryService that is used to communicate with other classes and objects. -streamer: Streamer : Streamer object of the library
Functions
+preparePresentation(publication : Publication, book : Book) :This function prepares a presentation with a given book and publication

3.3 Reader

ReaderModule

class ReaderModule
The logic of the reader package
Properties
-delegate : ReaderModuleDelegate : Delegate of the reader module, that is used to communicate with other objects within the program -books : BookRepository : Books of the user that are used, removed or added with the help of BooksRepository Class, that wraps db functions -bookmarks: BookMarkRepository -formatModules : ReaderFormatModule[] -factory : ReaderFactory(): factory is of type ReaderFactor, which has a number of useful properties and essential functions, it is used for wrapping.
Functions
+presentPublication(publication : Publication) : This is used to present a publication with passing the publication as a parameter

ReaderFactory

class ReaderFactory
Creates the reader module
Properties
none
Functions
+make() : ReaderModule: Wrapper function that creates the reader module

EPUB Package

Following functions will be added inside the EPUB package.

EPUBModule

class EPUBModule
Creates the viewcontroller for the operations of the delegate objects.
Properties
-publicationFormats : MediaFormat[] : It holds the publication formats inside a MediaFormat type array +makeReaderViewController(publication : Publication, locator : Locator, bookId : Book.Id, books : BookRepository, bookmark : BookmarkReposity) : UIViewController: This function is used to create a ReaderViewController, given the required the parameters.
Functions
none

EPUBViewController

class EPUBViewController
Presents the books and the audiobook and in the model shows the audiobooks and book using the library service
Properties
-toolbar: UIToolBar :Toolbar of the view -goToPageButton: UIButton :Go to page button, that is used to open the desired page given the page number -pageSlider: UISlider :Page slider, that is used slide through the pages, with the use of a slider -popUpView: UIView :This property is used for the popUpViews of the program, that usually opens from the bottom of the page -epubNavigator: EPUBNavigatorViewController : Epub navigator that is of type EPUBNavigatorViewController -playButton: UIButton : Play button that is used for audiobooks or for the narration mode -mode: NavigatorMode : Mode of the navigator -forwardButton: UIButton : Forward button that is used for audiobooks or for the narration mode
Functions
+hideToolBar() : Hides the toolbar init(publication: Publication, locator: Locator?, bookId: Book.Id, books: BookRepository, bookmarks: BookmarkReposity, resourcesServer: ResourcesServer) : Initiates the view with given the required parameters +changeMode(mode: NavigatorMode) : Changes the mode of the view with the given mode passed as a parameter

PopUpViewController

class PopUpViewController
Presents the popups and with a model. views the popups.
Properties
-type: PopUpTypes : Type of the popup -stackView: UIStackView : This is used for making the pop up stackview -scrollView: UIScrollView : This is used for making the pop up scrollview
Functions
+hideToolBar() : Hides the toolbar init(publication: Publication, locator: Locator?, bookId: Book.Id, books: BookRepository, bookmarks: BookmarkReposity, resourcesServer: ResourcesServer) : Initiates the view with given the required parameters +changeMode(mode: NavigatorMode) : Changes the mode of the view with the given mode passed as a parameter +addPopUps() : Used for adding popup +ResetStackView() :Used for resetting the popup +addNewView() : Used for adding a new view

3.4 Flashcard

DeckCollectionViewController

class DeckCollectionController
Models and views the decks
Properties
-decks: Deck[] : Array of decks that are shown in the view
Functions
+createDeck(name: String) :This function is used for creating a deck given a name(a string) +removeDeck(deckId: Id) : This function is used for removing a deck from deck collection +openDeck(deckId: Id) :This opens the deck with passing the ID as a parameter into the function

FlashCardViewController

class FlashCardViewController
Model the logic behind it and views the flashcards
Properties
-deck: Deck :The deck of the flashcard -progress: Double :Progress of the flashcard -okButton: UIButton :Button that is used to confirm the card -minusButton: UIButton : Button that is used to not to confirm the card -notrButton: UIButton : Button that is used when the user is not sure where the card is to be confirmed or not-
Functions
+showNextCard() :It is used to show the next card on the deck +removeCard() :It is used to remove the selected card from the deck +editCard(card: FlashCard) :This function lets user to edit the card with passing the flashcard as a parameter with type card

4. References

[1]Institute, Mpower A.S.C.E.R.T. “Language Learning Leads to Improved Personality.” ASCERT, ASCERT, 29 Jan. 2020, https://www.ascertinstitute.org/post/language-learning-leads-to-improved-personality.

[2]Gobler, E., 2021. Best language learning app 2021: Top 7 apps compared | ZDNet. [online] ZDNet. Available at:

<https://www.zdnet.com/article/best-language-learning-app/#:~:text=Duolingo%20has%20become%20the%20most,methods%20to%20help%20you%20learn.> [Accessed 20 December 2021].

[3]“Why I’m Quitting the Japanese Duolingo Course (an Honest Review).” Www.youtube.com, www.youtube.com/watch?v=jf-SbSfiXn4&t=480s&ab_channel=Livakivi. Accessed 20 Dec. 2021.

[4]Wikipedia contributors. "Kató Lomb." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 15 Nov. 2021. Web. 20 Dec. 2021.

[5] Oracle.com. 2022. *What is a database?*. [online] Available at: <https://www.oracle.com/database/what-is-database/> [Accessed 26 February 2022].

[6] 2022. [online] Available at: <https://www.merriam-webster.com/dictionary/flash%20card> [Accessed 27 February 2022].

[7] Tutorialspoint.com. 2022. *Computer Programming - Functions*. [online] Available at: <https://www.tutorialspoint.com/computer_programming/computer_programming_functions.htm> [Accessed 27 February 2022].