Project Booxchange (BookExchange)

Addis Ababa Institute of Technology

School of Information Technology and Engineering

Object-Oriented Programming - Project Work Proposal

Group Members' Names

Roll No	Name	ID Number
1	Biniyam Seid	UGR/9483/13
2	Fikernew Birhanu	UGR/9932/13
3	Leul Degarege	UGR/5038/13
4	Lidya Ejigu	UGR/6806/13
5	Yeabsira Yetwale	UGR/8664/13
6	Yohannes Ejigu	UGR/3361/13

Submitted to: Michael Sheleme Beyene

TABLE OF CONTENTS

1. Introduction

Motivation:

Reading is a good habit many try to develop. With the price of books soaring many cannot afford to buy all of the books they want to read. Also, we all have books that have been on our shelves for too long. Books from lower grades or others that aren't typically read again. And we don't have the books we would like to read.5 This creates a scenario in which these books don't have much use after the first time, therefore, as a result, are deemed surplus that just essentially clutter up space after that first use. Also, there is this way of lifestyle that really captured our attention, minimalism. The concept of minimalism is essentially that a person should have just about enough to survive, and all rest just doesn't make us a better person. It says we should not amass items or things just because, and every item in one's belongings should have a purpose for that person. This brings us to the purpose of our project. To make all books not just things that take up space, but also have purpose and be actively used.

This app will let its users exchange books with readers in your area. If you have one book you can exchange it to read another book.

Related Systems:

This app relies on its database to store login credentials of its users and admins. Along with credentials it stores books that are available to be exchanged. And it continuously updates itself as books are exchanged and are no longer available. The admins have the privilege of removing books that are deemed inappropriate. They can also remove/suspend users based on reports. The database stores the location of its users and data the app relies on. To achieve that it relies on the mobiles of its users to provide the location. The app might request its users from time to time to update their current location. Because it helps them get better recommendations based on their address.

The problem the app is trying to solve is:

This app provides the platform for users to be able to enlist and exchange books with other people who want to change books, as well. So functionally, if successfully implemented, this app will allow users to find and match with other users who are interested in changing with them.

2. Scope

Description:

Why bother with already read books cluttering space at home? With this android app, one can create accounts and instantly find a new home for their old books and a new book to read themselves!

The app will:

- Allow users to enlist their books
- Provide a way for users to interact with each other
- Suggest potential matches based on geographical proximity and interests for the users
- Allow new users to sign up
- Show on-demand books
- Provide communication and commenting platform for uses
- Provide different ways for users to change their belongings
- Allow users to request and offer auctions for certain books of their interests
- have a place for specifically educational books
- Run ads to support running costs

The app will not:

- Provide a market place to buy books
- Not support rich communication (that is it will not accept large data types such as Videos, zips,...)
- Store books in any shape or form (especially Ebooks)
- Provide in-app billing and buying services

3. Users and Roles:

The users of this app and their roles are:

Users:

- Enlist the books they would like to change
- Offer for (request) books they would like to have
- Rate other users for their excellent bookmanship

- Report other users for inappropriate behavior
- Sign-up and log in using their credentials
- Provide current locations for better experience
- Ask assistance from admins

Administrators:

- Approve of new book listings and requests for books
- Accept and manage reports
- Provides support for admins when asked.
- Monitor currently enlisted books and manage them.

4. Functional Requirements:

ID	Name	Description	Dependencies
FR01	Sign Up	App must allow new users to register	-
FR02	Log in	App must allow registered users to login	-
FR03	Enlisting	App must allow users to add their book to the enlisted books list	FR02
FR04	Search	App must allow users to search the database for items using book titles and generate a list of books in response to the search	FR02
FR04	Request	App must allow users to request books of their choices	FR02

ID	Name	Description	Dependencies
FR01	Sign Up	App must allow new users to register	-
FR02	Log in	App must allow registered users to login	-
FR05	Report	App must allow users to report inappropriate behavior	FR02
FR06	Removing	App must allow admins to remove specific books from the database	FR02
FR07	Completions signaling	App must allow for users to signal the completion of a swap (barter trade)	FR03
FR08	Suggestion generation	App must suggest books based on age groups	FR02

5. Design:

Use Case Scenarios

1. Enlist a book

- user makes a book available by specifying *book name, author name, year of publication, years spent with the user, genre of the book.*
- administrators receive the enlisting request
- administrators approve the enlisting
- database handler adds enlisted book to the database

2. Request a book

- user makes a request for a book by specifying book name, author name, year of publication
- administrators receive the request
- administrators approve of the request
- database handler adds the request to the database

3. Search for a book

- user makes a search by using book name or author name
- database handler receives the parameters of the search
- database handler queries the database for the searched items
- database returns the results of the search
- database handler displays the search results to the user

4. Sign up

- user enters the app and clicks sign up button
- user fills a form regarding name, age, gender, email, phone, location
- database handler receives the request and checks for other entries in the database with similar unique values
- database handler returns if there exists a similar user
- Database checks and green lights the sign up
- database handler adds the user to the database
- the user is signed up and can log in

5. Log in

- user enters username/phone and password
- database handler receives the request and checks the database with the keys given by the user
- database handler returns a result if the given credentials exist
- database handler grants a session
- a user profile is created containing all information about the user from the database
- the user is logged in

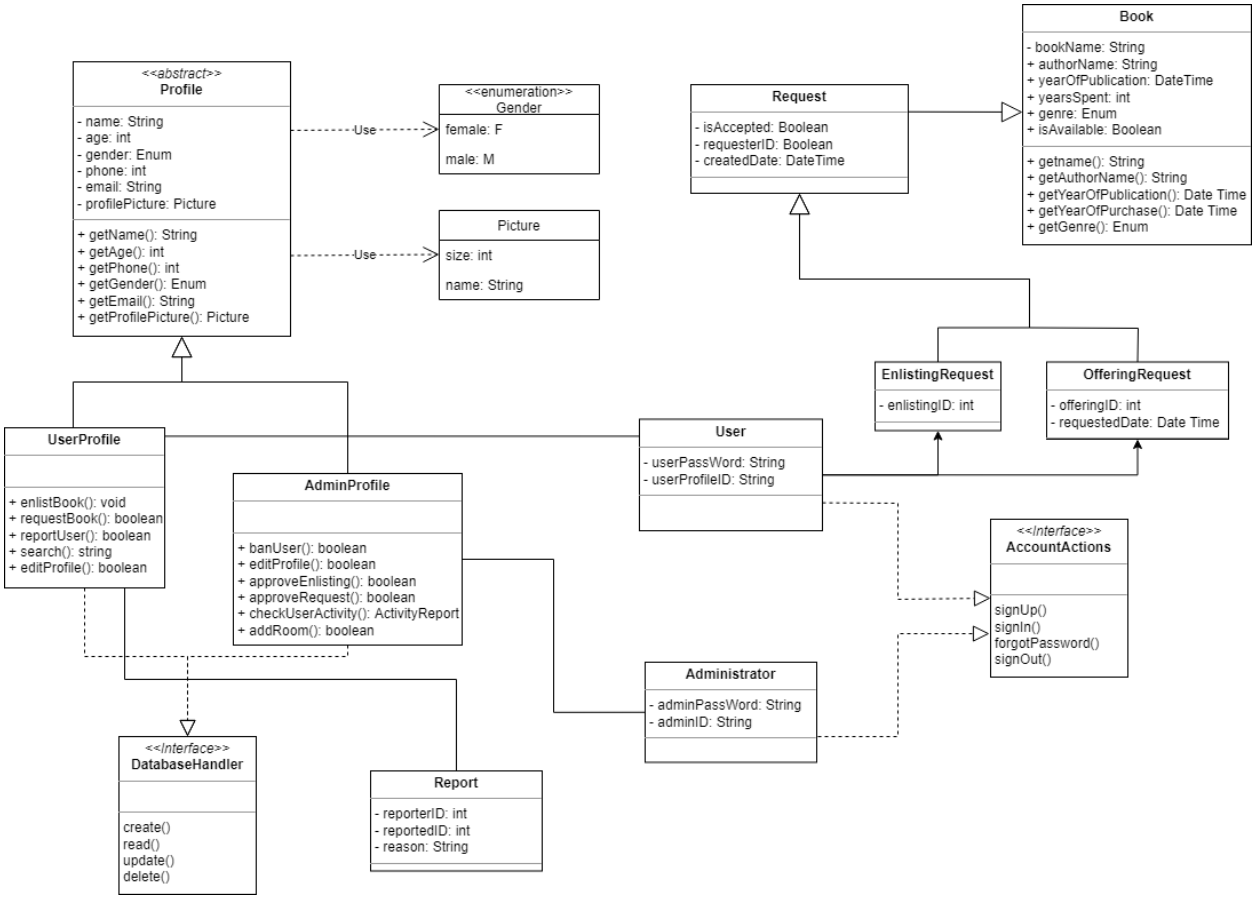
6. Report inappropriate behavior

- user reports inappropriate behavior by specifying the username of the reported/abuser
- administrator receives the report request
- administrators check recent activities of the alleged abuser
- administrators approve of the report
- abuser is banned from the system

7. Edit profile information

- User tries to change one or more qualities of their profile
- Database handler receives the request
- Database handler queries for the change is the intended change is allowed or not
- Database greenlights the change
- Profile information is updated on the database

UML Diagram



Classes

1. <<Abstract>> Profile

This is an abstract class that will be inherited by different kinds of users the app might have. Currently, the app has only two sections of users, namely, Users and Administrators but then if other roles are to emerge, this class will make sure those protocols are conformed to. Objects created from this app will instantly receive their attributes from their respective classes from the database.

Attributes

Name	**Type**	**Visibility**
name	string	private
age	Int	private
gender	enum	private
phone	int	private
email	String	private
ProfilePicuture	Picture	private

Methods

Name	**Type**	**Visibility**
getName()	String	public
getAge()	int	public
getPhone()	int	public
getGender()	Enum	public
getEmail()	String	public
getProfilePicture()	Picture	public

2. UserProfile

This is a user class that any user will be assigned based on the credentials they enter into the application’s form. It is through this app's methods users will be able to use the app. That is, this class will inherit from the Profile class which means it will be able to explain users using different attributes to be used to make the UI interactive whilst also having robust methods that make it to achieve its goals.

Attributes

Name	**Type**	**Visibility**
userID	String	private
userPassword	String	private
userProfileID	String	private

Methods

* Implements AccountActions

3. AdministratorProfile

Like the class before it, AdminstratorProfile is a class that represents the Administrator and performs the required functionalities for admins.

Attributes

Name	**Type**	**Visibility**
adminID	String	private
adminPassword	String	private
adminProfileID	String	private

Methods

* Implements AccountActions

4. <<interface>> AccountActions

This is an interface that will hold methods related to account credential manipulations. A class implementing this object will instantly gain access to methods that perform actions such as signing in, signing up, and the like. All in all, this interface will handle all queries towards the database up until the user successfully logs in and objects of the profile classes take over.

Methods

Name	**Type**	**Visibility**
signUp()	Void	Private
signIn()	Void	Private
forgotPassword()	Void	Private
signOut()	Void	Private

5. Book

This is a class that books will be assigned when users claim. It is through this class's methods users will be able to get info about books they can exchange.

Attributes

Name	**Type**	**Visibility**
bookName	String	private
authorName	String	public
yearOfPublication	DateTime	public
yearsSpent	int	public
genre	Enum	public
isAvailable	Boolean	public

Methods

Name	Type	Visibility
getname()	String	private
getAuthorName()	String	private
getYearOfPublication()	String	private
getYearOfPurchase()	String	private
getGenre()	String	private

6. EnlistingRequest

Not every book a user requests to exchange is available to exchange; it should wait for approval from the admin. This class manages those requests for approval.

Attributes

Name	Type	Visibility
enlistingID	int	private

7. Administrator

This is a class for admins and they will be assigned an object of this class based on the credentials they enter into the application's form.

Attributes

Name	Type	Visibility
administratorID	int	private
administratorPassWord	String	private

Methods

Name	Type	Visibility

banUser()	void	public
editProfile()	void	public
approveEnlisting()	void	public
approveRequest()	void	public
checkUserActivity()	void	public
addRoom()	void	public

8. <<interface>> DatabaseHandler

This is the interface that will be a kind of bridge between our application and the database and handle all queries to the database to perform all CRUD operations that keep the application performing. The main actors in the application, namely the objects of the classes UserProfile and AdministratorProfile will implement this interface and adapt its methods to properly connect and manipulate the external database which will store all the data concerning the application.

Methods

Name	**Type**	**Visibility**
create()	void	private
read()	void	private
update()	void	private
delete()	Void	private

9. ExternalDatabase

This is a generic class showing that there will be links with an outside database that will be used to store all the data of the social media.

10. OfferingRequest

This is a class that allows users to make requests about the books they are looking for. If anyone happens to be willing to exchange requests the app will connect the two users.

Attributes

Name	Type	Visibility
bookName	String	private
authorName	String	public
requestedDate	String	public

Methods

11. Report

Through this class users can reach admins. They can report inappropriate books or users. The class has attributes that will allow the admin to learn about the reason for the report.

Attributes

Name	Type	Visibility
reporterID	int	private
reportedID	int	private
reason	String	private

References

1. UML Class Diagram Tutorial, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/, 2022
2. Suzanne Robertson, James Robertson, Mastering the Requirements Process: Getting Requirements Right, 2012
3. Karl Wiegers, Joy Beatty, Software Requirements, 2013
4. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, 1994
5. Robert C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, 2017