Project Specification:
****Library Management System****

****Objective:****
The objective of this project is to develop a library management system that allows users to interact with the library by borrowing items, attending events, donating items, and seeking assistance from librarians. The system will be implemented using Python's SQLite3 library to connect to a database populated using SQL.

****Functionalities:****
1. User Management:
   - User Registration: Users can register with the system by providing a unique username, email, password, and user type (e.g., member, librarian, volunteer).
   - User Login: Registered users can log in to the system using their credentials.

2. Item Management:
   - Add Item: Librarians can add new items to the library, including print books, online books, magazines, CDs, and records. Each item should have a unique identifier (ItemID), item type (ItemType), title, author (for books), publisher (for books and magazines), and availability status (Availability).
   - Search Item: Users can search for specific items in the library based on title, author, item type, etc.
   - Borrow Item: Users can borrow available items from the library. The system should check the availability status and record the borrowing details, including the user who borrowed the item, borrow date, and due date.

3. Event Management:
   - Add Event: Librarians can add new events to the library, such as book clubs, art shows, and film screenings. Each event should have a unique identifier (EventID), event type (EventType), title, description, date, location, and target audience (Audience).
   - Search Event: Users can search for upcoming events at the library based on event type, title, date, etc.
   - Attend Event: Users can register for events they want to attend. The system should keep track of event attendees.

4. Donation Management:
   - Donate Item: Users can donate items (e.g., books, CDs, records) to the library. The donated items should be added to the library's collection.

5. Personnel Management:
   - Add Personnel: Administrators can add personnel (e.g., librarians, volunteers) to the system. Each personnel should have a unique identifier (PersonnelID), name, and position.

6. Help Desk:
   - Request Assistance: Users can request assistance from librarians regarding any library-related queries or issues.

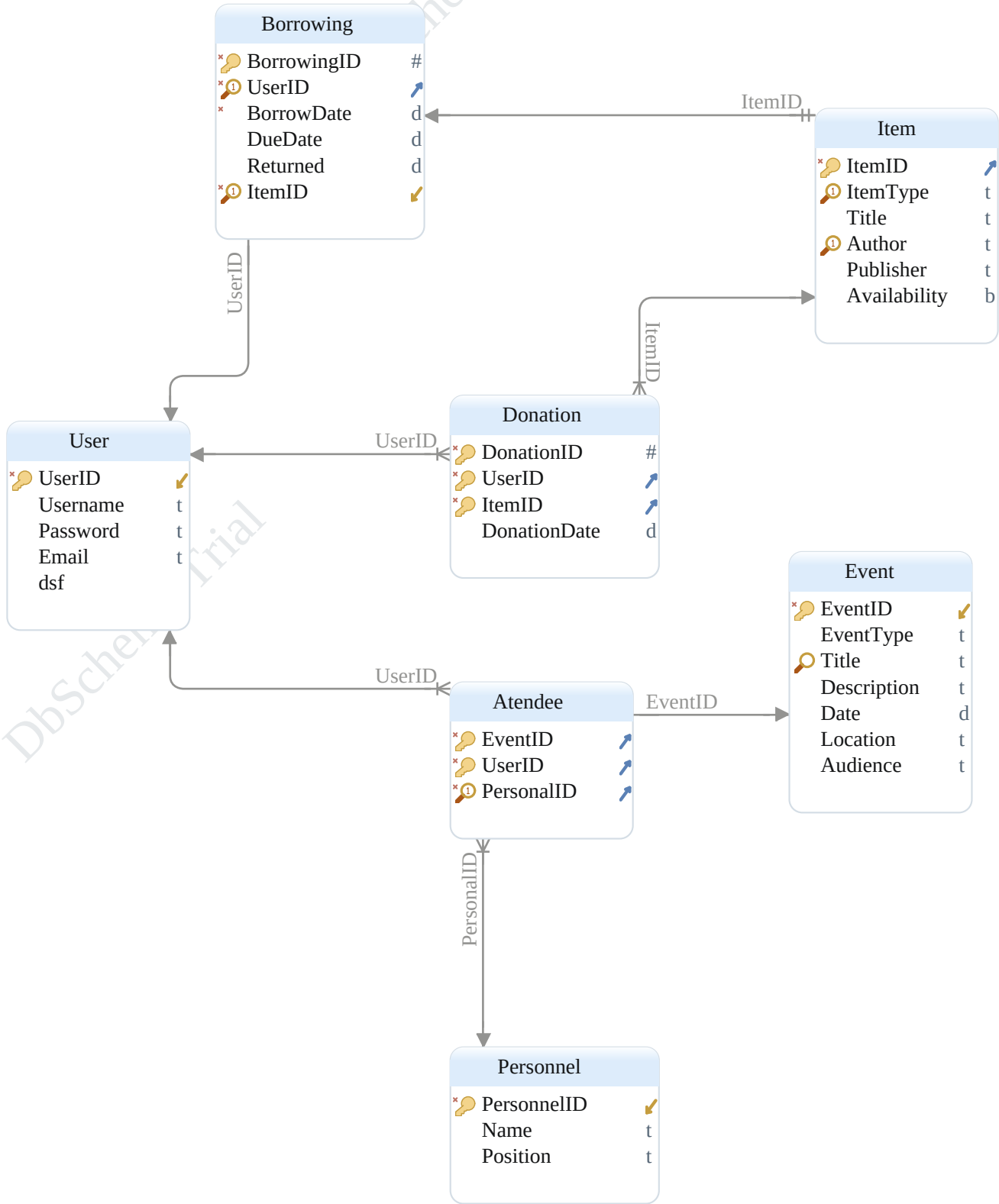
****Python and SQLite3 Implementation:****
1. Python will be used to create the user interface for the library management system. The system will use the SQLite3 library to connect to the SQLite database.

2. The Python code will handle user authentication, user input validation, and database interactions for various functionalities, such as adding items, events, personnel, and donations, searching for items and events, registering for events, borrowing items, etc.

3. SQL queries will be used to perform database operations, such as creating tables, inserting data, updating records, and retrieving information.

4. The Python code will utilize functions and classes to encapsulate the logic for different operations, ensuring code reusability and maintainability.

****Note:****
- The overall project specification provided here serves as a high-level overview. The actual implementation may involve additional functionalities, error handling, and security measures to ensure data integrity and user privacy.


Borrowing
BorrowingID #
UserID
BorrowDate d
DueDate d
Returned d
ItemID
ItemID
Item
ItemID
ItemType t
Title t
Author t
Publisher t
Availability b
UserID
ItemID
User
UserID
Username t
Password t
Email t
dsf
UserID
Donation
DonationID #
UserID
ItemID
DonationDate d
Event
EventID
EventType t
Title t
Description t
Date d
Location t
Audience t
UserID
EventID
Atendee
EventID
UserID
PersonalID
PersonalID
Personnel
PersonnelID
Name t
Position t

Default Layout

Entity Atendee

* Pk	EventID	INT
* Pk	UserID	INT
* Unq	PersonalID	INT
Indexes		
Pk	pk	EventID, UserID
Unq	unq_PersonalID	PersonalID
Relationships		
	fk_Atendee_Event (EventID) ref Event (EventID)	
	fk_Atendee_User (UserID) ref User (UserID)	
	fk_Atendee_Personnel (PersonalID) ref Personnel (PersonnelID)	

Entity Borrowing

* Pk	BorrowingID	INT
* Unq	UserID	INT
*	BorrowDate	DATE
	DueDate	DATE
	Returned	DATE
* Unq	ItemID	INT
Indexes		
Unq	unq_ItemID	ItemID
Pk	pk_Borrowing	BorrowingID
Unq	unq_UserID	UserID
Relationships		
	fk_Borrowing_User (UserID) ref User (UserID)	

Entity Donation

* Pk	DonationID	INT
* Pk	UserID	INT
* Pk	ItemID	INT
	DonationDate	DATE
Indexes		
Pk	pk_Donation	DonationID, UserID, ItemID
Relationships		
	fk_Donation_Item (ItemID) ref Item (ItemID)	
	fk_Donation_User (UserID) ref User (UserID)	

Entity Event

* Pk	EventID	INT
	EventType	VARCHAR(100)
Idx	Title	VARCHAR(100)
	Description	VARCHAR(100)
	Date	DATE
	Location	VARCHAR(100)
	Audience	VARCHAR(100)
Indexes		
Pk	pk_Event	EventID
	idx_Event_Title	Title

Entity Item

* Pk	ItemID	INT
Unq	ItemType	VARCHAR(100)
	Title	VARCHAR(100)
Unq	Author	VARCHAR(100)
	Publisher	VARCHAR(100)
	Availability	BOOLEAN
Indexes		
Pk	pk_Item	ItemID
Unq	unq_Author	Author
Unq	unq_ItemType	ItemType
Relationships		
	fk_Item_Borrowing (ItemID) ref Borrowing (ItemID)	

Entity Personnel

* Pk	PersonnelID	INT
	Name	VARCHAR(100)
	Position	VARCHAR(100)
Indexes		
Pk	pk_Personnel	PersonnelID

Entity User

* Pk	UserID	INT
	Username	VARCHAR(100)
	Password	VARCHAR(100)
	Email	VARCHAR(150)
	dsf	SUBENTITY
Indexes		
Pk	pk_User	UserID