

COMPUTER SCIENCE PROJECT WORK

LIBRARY MANAGEMENT SYSTEM

Made By:

Manas Madan : 20

Sanyam Singh : 35

Uday Kalra : 41

CERTIFICATE

This is to certify that

Manas Madan, Sanyam Singh &
Uday Kalra of class 12 - C have
successfully completed the Library
Management Project under the
Guidance of Mrs. Nutan Parashar.

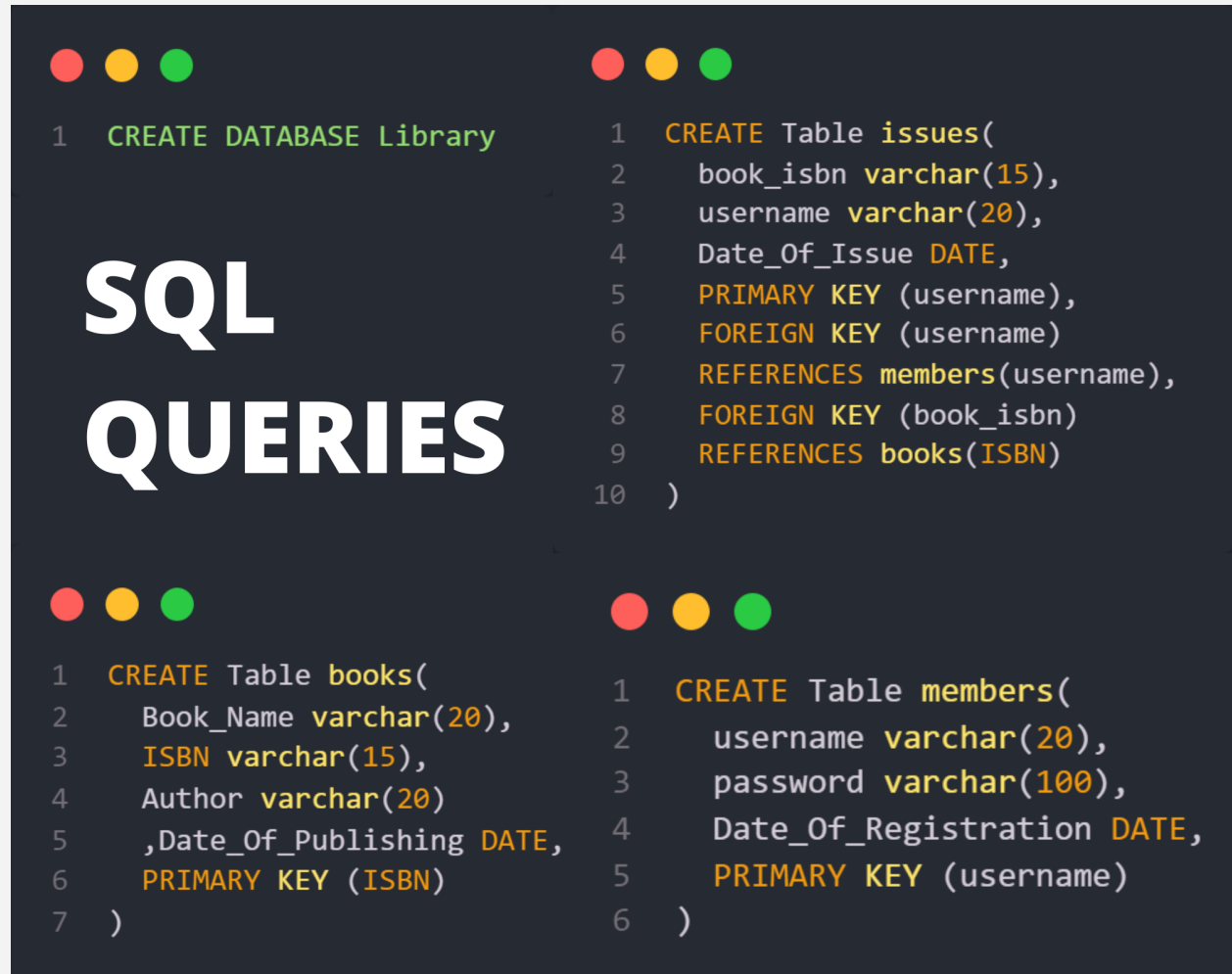
Acknowledgement

In the accomplishment of this project successfully we would like to express our special thanks to our Computer Science teacher

Mrs. Nutan Parashar for their able guidance and support in completing of our project

Date. 28 12 2021

SQL QUERIES USED:



```
1 CREATE DATABASE Library
```

SQL QUERIES

```
1 CREATE Table issues(  
2     book_isbn varchar(15),  
3     username varchar(20),  
4     Date_Of_Issue DATE,  
5     PRIMARY KEY (username),  
6     FOREIGN KEY (username)  
7     REFERENCES members(username),  
8     FOREIGN KEY (book_isbn)  
9     REFERENCES books(ISBN)  
10 )
```

```
1 CREATE Table books(  
2     Book_Name varchar(20),  
3     ISBN varchar(15),  
4     Author varchar(20)  
5     ,Date_Of_Publishing DATE,  
6     PRIMARY KEY (ISBN)  
7 )
```

```
1 CREATE Table members(  
2     username varchar(20),  
3     password varchar(100),  
4     Date_Of_Registration DATE,  
5     PRIMARY KEY (username)  
6 )
```

A Separate File [setup.py](#) has been created that does all this for you.

1. It Creates The Database Library
2. Then It Creates Three Tables Under Database Library:
 - a. Books
 - b. Members
 - c. Issues

BOOKS

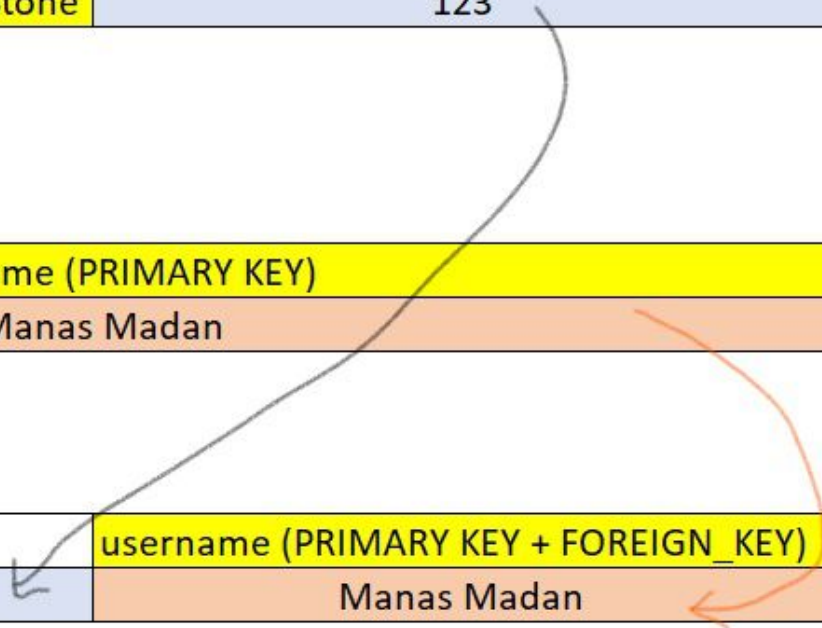
Book_Name (PRIMARY KEY)	ISBN	Author	Date_Of_Publishing
Harry Potter and the Philosopher's Stone	123	JK Rowling	26-06-1997

MEMBERS

username (PRIMARY KEY)	password	Date_Of_Registration
Manas Madan	password	21-06-2021

ISSUES

book_isbn (FOREIGN KEY)	username (PRIMARY KEY + FOREIGN_KEY)	Date_Of_Issue
123	Manas Madan	22-12-2021



Python Code

sqlData.py



```
1  DATABASE_NAME = "library"
2  # Edit The Variables below TO Run THE Code In Your Computer
3  # Replace root with your sql username [DEFAULT SQL Username is root]
4  SQL_USERNAME = "root"
5  SQL_PASSWORD = "root" # Replace root with your sql password
```


setup.py



```
1  # This is A One Time Run File After Running It You May Delete It
2  # This set ups the environment required to run the python program successfully
3  # Just Make Sure You Do Not Have Any Database name Library in your pc beforerunning this file
4
5  # Sql Connector
6  import mysql.connector
7  # SQL Data File
8  import sqlData
9
10 # Establishing The Connection
11 mydb = mysql.connector.connect(
12     host="localhost",
13     user=sqlData.SQL_USERNAME,
14     password=sqlData.SQL_PASSWORD
15 )
16
17 # Creating The Cursor Object
18 mycursor = mydb.cursor()
19 # Executing Commands
20 # Creating Database Library
21 mycursor.execute("CREATE DATABASE Library")
22 # Swithing To Database Library
23 mycursor.execute("use Library")
24
25 # Creating Tables :
26 # Books
27 mycursor.execute(
28     "CREATE Table books(Book_Name varchar(20),ISBN varchar(15),Author varchar(20),Date_Of_Publ\
29     ishing DATE,PRIMARY KEY (ISBN))")
30 # Members
31 mycursor.execute(
32     "CREATE Table members(username varchar(20),password varchar(100),Date_Of_Registration DATE,\
33     PRIMARY KEY (username))")
34 # Issues
35 mycursor.execute("CREATE Table issues(book_isbn varchar(15),username varchar(20),Date_Of_Issue \
36     DATE,PRIMARY KEY (username),FOREIGN KEY (username) REFERENCES members(username),FOREIGN KEY \
37     (book_isbn) REFERENCES books(ISBN))")
```

utils.py



```
1  # *****Imports*****
2  # OS Module To Implement The Clear Screen Function
3  import os
4  # MySQL Connector
5  import mysql.connector
6  from mysql.connector import errorcode
7  from mysql.connector import (connection)
8  # bcrypt Library to hash passwords
9  import bcrypt
10 # Local Python File sqlData containing the sql Username ,Password and Database Name
11 import sqlData
12
13
14 # *****General Functions*****
15 # Class Colors to print clorful text in python terminal
16 class Colors:
17     HEADER = '\033[95m'
18     OKBLUE = '\033[94m'
19     OKCYAN = '\033[96m'
20     OKGREEN = '\033[92m'
21     WARNING = '\033[93m'
22     FAIL = '\033[91m'
23     ENDC = '\033[0m'
24     BOLD = '\033[1m'
25
26
27 # MenuSpacing Variable To Center The menu Heading by adding spaces as prefix
28 menuSpacing = "          "
29
30 # Function To Print Divider in Green Color
31
32
33 def divider():
34     print(f"{Colors.OKGREEN}===== {Colors.ENDC}")
```



```
1  # Function To Print Invalid Input in Red Color
2
3
4  def invalidInput():
5      print(f"{Colors.FAIL}Invalid Input Enter A Choice From the Menu Above{Colors.ENDC}")
6      divider()
7
8
9  def clearScreen():
10     os.system("cls")
11
12
13  # *****Menu Functions*****
14  # Function To Print Menu
15  def showMenu(menuTitle, menuOptions, menuSpacing):
16      while True:
17          print(f"{Colors.HEADER}{menuSpacing}{menuTitle} {Colors.ENDC}")
18          divider()
19          for choice, statement in menuOptions.items():
20              print(
21                  f"{Colors.OKBLUE}{Colors.BOLD}{choice} - {statement[0]}{Colors.ENDC}")
22          try:
23              choice = int(input("Enter Choice : "))
24              clearScreen()
25              if(choice >= 1 and choice <= len(menuOptions.keys())):
26                  if(choice == len(menuOptions.keys())):
27                      break
28                  try:
29                      menuOptions[choice][1]()
30                  except Exception as e:
31                      print(e)
32                      divider()
33              else:
34                  invalidInput()
35          except:
36              clearScreen()
37              invalidInput()
```



```
1  # *****SQL Functions Start*****
2  # SQL Function To Add/Delete and Update Data in the database
3  # NOTE : Cannot Be Used To Read Data From Cursor Object as Connection is Broken when Function Ends
4  def executeSQLCommitQuery(query, data):
5      try:
6          cnx = connection.MySQLConnection(
7              user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD, host='localhost',
8              database=sqlData.DATABASE_NAME)
9          Cursor = cnx.cursor()
10         Cursor.execute(query, data)
11         cnx.commit()
12         Cursor.close()
13         cnx.close
14         return Cursor
15     except mysql.connector.Error as err:
16         return handleSQLException(err)
17
18 # SQL Function To Handle ALL Major SQL ErrorCodes else return error message
19
20
21 def handleSQLException(err):
22     if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
23         return ("Something is wrong with your user name or password")
24     elif err.errno == errorcode.ER_BAD_DB_ERROR:
25         return ("Database does not exist")
26     elif err.errno == 1062:
27         return ("Duplicate Entry")
28     elif err.errno == 1451:
29         return ("Cannot Delete A Entry that Is A Member Of Another Table, the book or user must be\
30             issued to someone try removing the issued record first")
31     else:
32         return (err)
```



```
1  # *****Password Functions*****
2  # Hash Password String After Adding Salt using bcrypt
3  def hashPassword(password):
4      return bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())
5
6  # Compare Password Strings using bcrypt
7
8
9  def checkPassword(password, hashedPassword):
10     return bcrypt.checkpw(password.encode("utf-8"), hashedPassword.encode("utf-8"))
```

main.py



```
1  # Importing ALL Functions from Utils Python File
2  from utils import *
3  # Importing Books Python File
4  import books
5  # Importing Members Python File
6  import members
7  # Importing Issues/Return Python File
8  import issueReturn
9
10 # Running The Below Code Only If The File Is Not Been Imported
11 if __name__ == "__main__":
12     # Creating The Menu Options Dictionary
13     mainMenuOptions = {
14         1: ["Book Management", books.menu],
15         2: ["Members Management", members.menu],
16         3: ["Issue / Return Book", issueReturn.menu],
17         4: ["Exit"]
18     }
19     # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
20     # Passing Main Menu as Menu Title and Menu Spacing from utils file to center the main menu heading
21     showMenu("Main Menu", mainMenuOptions, menuSpacing)
```

books.py



```
1  # Importing ALL Functions From The utils file
2  from utils import *
3  # Importing datetime module
4  from datetime import date
5  # Importing PrettyTable from prettyprints module to print tables
6  from prettytable import PrettyTable
7  # Importing sqlData
8  import sqlData
9
10
11 def addBook():
12     # Try Catch Block
13     try:
14         # Taking User Inputs
15         BOOK_NAME, BOOK_ISBN, BOOK_AUTHOR, BOOK_PUBLISHING_DATE,
16         BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_YEAR = \
17         input(f"{Colors.OKCYAN}Enter Book Name : "),
18         int(input("Enter Book ISBN : ")),
19         input("Enter Book Author : "),
20         int(input("Enter Book Publishing Date : ")),
21         int(input("Enter Book Publishing Month : ")),
22         int(input(f"Enter Book Publishing Year : {Colors.ENDC}"))
23
24         # Executing SQL Query
25         query = "INSERT INTO books VALUES (%s, %s, %s, %s);"
26         data = (BOOK_NAME, BOOK_ISBN, BOOK_AUTHOR, date(
27             BOOK_PUBLISHING_YEAR, BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_DATE))
28         # Executing The Query From Function in utils file
29         res = executeSQLCommitQuery(query, data)
30         # Clearing Screen For Better Presentation
31         clearScreen()
32         # If the result comes as a string it means a error has been raised else Member Get Added
33         if(type(res) == str):
34             print("Error : ", res)
35         else:
36             print("Book Added")
37     except:
38         clearScreen()
39         print("Invalid Input")
```



```
1  def displayBooks():
2      # Try Catch Block
3      try:
4          # Executing SQL Query
5          query = "select * from books;"
6          cnx = connection.MySQLConnection(
7              user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
8              host='localhost', database=sqlData.DATABASE_NAME)
9          Cursor = cnx.cursor()
10         Cursor.execute(query)
11
12         # Creating PrettyTableObject
13         x = PrettyTable()
14         # Adding Field Names
15         x.field_names = ["Book Name", "Book ISBN",
16                         "Book Author", "Date Of Publishing"]
17         for i in Cursor:
18             # Using Datetime strftime to convert datetime object to string
19             dateString = date.strftime(i[3], "%d-%m-%Y")
20             x.add_row([i[0], i[1], i[2], dateString])
21
22         # Printing The Table
23         print(x)
24
25         # Closing the Connection
26         Cursor.close()
27         cnx.close()
28
29     except Exception as err:
30         print(handleSQLException(err))
```



```
1  def searchBook():
2      # Try Catch Block
3      try:
4          # Taking User Input
5          BOOK_ISBN = int(
6              input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))
7
8          # Executing SQL Query
9          query = 'select * from books where ISBN = "%s";'
10         data = (BOOK_ISBN,)
11         cnx = connection.MySQLConnection(
12             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
13             host='localhost', database=sqlData.DATABASE_NAME)
14         Cursor = cnx.cursor()
15         Cursor.execute(query, data)
16
17         # Creating PrettyTableObject
18         x = PrettyTable()
19         # Adding Field Names
20         x.field_names = ["Book Name", "Book ISBN",
21                         "Book Author", "Date Of Publishing"]
22
23         # Creating Found Variable
24         found = False
25         # Navigating In Cursor Object To Add Values TO PrettyTable Object
26         for i in Cursor:
27             # Using Datetime strftime to convert datetime object to string
28             dateString = date.strftime(i[3], "%d-%m-%Y")
29             data = [i[0], i[1], i[2], dateString]
30             x.add_row(data)
31             # Changing Found Variable TO True
32             found = True
33
34         # If Not Found Print Not Found
35         if(not found):
36             x.add_row(["No", "Book", "Record", "Found"])
37
38         # Printing The Table
39         print(x)
40         # Closing the Connection
41         Cursor.close()
42         cnx.close()
43
44     except Exception as err:
45         try:
46             print(handleSQLException(err))
47         except:
48             print(err)
```



```
1  def deleteBook():
2      # try-Catch Blck
3      try:
4          # getting the user input
5          BOOK_ISBN = int(
6              input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))
7
8          # Executng The Query
9          query = 'DELETE FROM books where ISBN="%s"'
10         data = (BOOK_ISBN,)
11         # Executing The Query From Function in utils file
12         res = executeSQLCommitQuery(query, data)
13         # Clearing Screen
14         clearScreen()
15
16         # If the result comes as a string it means a error has been raised else Member Get Added
17         if(type(res) == str):
18             print("Error : ", res)
19         else:
20             print("Book Deleted")
21     except:
22         clearScreen()
23         print("Invalid Input")
```



```
1  def updateBook():
2      # Try Catch Block
3      try:
4          # Taking User Input
5          BOOK_ISBN = int(
6              input(f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"))
7
8          # Executing SQL Query
9          query = 'select * from books where ISBN = "%s";'
10         data = (BOOK_ISBN,)
11         cnx = connection.MySQLConnection(
12             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
13             host='localhost', database=sqlData.DATABASE_NAME)
14         Cursor = cnx.cursor()
15         Cursor.execute(query, data)
16
17         # Creating PrettyTableObject
18         x = PrettyTable()
19         # Adding Field Names
20         x.field_names = ["Book Name", "Book ISBN",
21                         "Book Author", "Date Of Publishing"]
22
23         # Creating Found Variable
24         found = False
25         for i in Cursor:
26             # Using Datetime strftime to convert datetime object to string
27             dateString = date.strftime(i[3], "%d-%m-%Y")
28             data = [i[0], i[1], i[2], dateString]
29             x.add_row(data)
30             # Changing Found Variable TO True
31             found = True
32
33         # Printing Error If Not Found
34         if(not found):
35             x.add_row(["No", "Book", "Record", "Found"])
36             print(x)
37             return
38
39         # Printing The Table
40         print(x)
41         # Closing the Connection
42         Cursor.close()
43         cnx.close()
```



```
1      # Getting User Input For New Details
2      BOOK_NAME,
3      BOOK_AUTHOR,
4      BOOK_PUBLISHING_DATE,
5      BOOK_PUBLISHING_MONTH,
6      BOOK_PUBLISHING_YEAR = \
7      input(f"{Colors.OKCYAN}Enter Book Name : "),
8      input("Enter Book Author : "),
9      int(input("Enter Book Publishing Date : )),
10     int(input("Enter Book Publishing Month : )),
11     int(input(f"Enter Book Publishing Year : {Colors.ENDC}"))
12
13     # Executing SQL Query
14     query = 'UPDATE books set Book_Name=%s,Author=%s,Date_Of_Publishing=%s where ISBN="%s"'
15     data = (BOOK_NAME, BOOK_AUTHOR, date(BOOK_PUBLISHING_YEAR,
16         BOOK_PUBLISHING_MONTH, BOOK_PUBLISHING_DATE), BOOK_ISBN)
17     # Executing The Query From Function in utils file
18     res = executeSQLCommitQuery(query, data)
19     # Clearing Screen For Better Presentation
20     clearScreen()
21
22     # If the result comes as a string it means a error has been raised else Member Get Added
23     if(type(res) == str):
24         print("Error : ", res)
25     else:
26         print("Book Record Updated")
27
28 except Exception as err:
29     try:
30         print(handleSQLException(err))
31     except:
32         print(err)
```




```
1  def menu():
2      # Creating The Menu Options Dictionary
3      booksMenuOptions = {
4          1: ["Add Book", addBook],
5          2: ["Display Books", displayBooks],
6          3: ["Delete Book", deleteBook],
7          4: ["Search Book", searchBook],
8          5: ["Update Book", updateBook],
9          6: ["Return To Main Menu"]
10     }
11     # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
12     # Passing Members Menu as Menu Title and Menu Spacing from utils file to center the menu heading
13     showMenu("Books Menu", booksMenuOptions, menuSpacing)
```

members.py



```
1  def menu():
2      # Creating The Menu Options Dictionary
3      membersMenuOptions = {
4          1: ["Add Member", addMember],
5          2: ["Display Members", displayMembers],
6          3: ["Search Members", searchMember],
7          4: ["Delete Member", deleteMember],
8          5: ["Update Member Password", updateMemberPassword],
9          6: ["Return To Main Menu"]
10     }
11     # Running The Show Menu Function From Utils File On The Main Menu
12     Options Dictionary
13     # Passing Members Menu as Menu Title and Menu Spacing from utils f
14     ile to center the menu heading
15     showMenu("Members Menu", membersMenuOptions, menuSpacing)
```



```
1  # Importing ALL Functions From The utils file
2  from utils import *
3  # Importing datetime module
4  from datetime import date
5  # Importing PrettyTable from prettyprints module to print tables
6  from prettytable import PrettyTable
7  # Importing sqlData
8  import sqlData
9
10 # Creating The Add Member Function
11 def addMember():
12     # Try Catch Block
13     try:
14         # Taking User Inputs
15         USER_NAME,
16         USER_PASSWORD,
17         DATE_OF_REGISTRATION = \
18         input(f"{Colors.OKCYAN}Enter User Name : "),
19         input(f"Enter User Password : {Colors.ENDC}"),
20         date.today()
21
22         # Checking If length of USER_NAME is less than 4 or USER_PASSWORD is less than 4
23         # If raising an Exception
24         if(len(USER_NAME) < 4 or len(USER_PASSWORD) < 4):
25             raise Exception(f"{Colors.FAIL}Invalid Input{Colors.ENDC}")
26
27         # Hashing User Password To Save it in sql table
28         USER_PASSWORD = hashPassword(USER_PASSWORD)
29
30         # Executing SQL Query
31         query = "INSERT INTO members VALUES (%s, %s, %s);"
32         data = (USER_NAME, USER_PASSWORD, DATE_OF_REGISTRATION)
33
34         # Executing The Query From Function in utils file
35         res = executeSQLCommitQuery(query, data)
36         # Clearing Screen For Better Presentation
37         clearScreen()
38         # If the result comes as a string it means a error has been raised else Member Get Added
39         if(type(res) == str):
40             print("Error : ", res)
41         else:
42             print("Member Added")
43     except:
44         clearScreen()
45         print("Invalid Input")
```



```
1  # Creating Display members Function
2  def displayMembers():
3      # Try-Catch Block
4      try:
5          # Writing SQL Query and Executing it
6          # The Function cannot be used here because Cursor object gets dele
7          ted from memory when function ends
8          query = "select * from members;"
9          cnx = connection.MySQLConnection(
10              user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
11              host='localhost', database=sqlData.DATABASE_NAME)
12          Cursor = cnx.cursor()
13          Cursor.execute(query)
14
15          # Creating PrettyTableObject
16          x = PrettyTable()
17          # Adding Field Names
18          x.field_names = ["User Name", "Date Of Registration"]
19
20          # Navigating In Cursor Object To Add Values TO PrettyTable Object
21          for i in Cursor:
22              # Using Datetime strftime to convert datetime object to string
23              dateString = date.strftime(i[2], "%d-%m-%Y")
24              x.add_row([i[0], dateString])
25
26          # Printing The Table
27          print(x)
28          # Closing the Connection
29          Cursor.close()
30          cnx.close()
31
32      except Exception as err:
33          print(handleSQLException(err))
```



```
1  def searchMember():
2      # try Catch Block
3      try:
4          # getting the user input
5          USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")
6
7          # Executng The Query
8          query = 'select * from members where username = %s;'
9          data = (USER_NAME,)
10         cnx = connection.MySQLConnection(
11             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
12             host='localhost', database=sqlData.DATABASE_NAME)
13         Cursor = cnx.cursor()
14         Cursor.execute(query, data)
15
16         # Creating PrettyTableObject
17         x = PrettyTable()
18         # Adding Field Names
19         x.field_names = ["User Name", "Date of Registration"]
20
21         # Creating Found Variable
22         found = False
23         # Navigating In Cursor Object To Add Values TO PrettyTable Object
24         for i in Cursor:
25             # Using Datetime strftime to convert datetime object to string
26             dateString = date.strftime(i[2], "%d-%m-%Y")
27             data = [i[0], dateString]
28             x.add_row(data)
29             # Changing Found Variable TO True
30             found = True
31
32         # If Not Found Print Not Found
33         if(not found):
34             x.add_row(["No User", "Record Found"])
35
36         # Printing The Table
37         print(x)
38         # Closing the Connection
39         Cursor.close()
40         cnx.close()
41
42     except Exception as err:
43         try:
44             print(handleSQLException(err))
45         except:
46             print(err)
```



```
1  def deleteMember():
2      # try-Catch Blck
3      try:
4          # getting the user input
5          USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")
6
7          # Executng The Query
8          query = 'delete from Members where username=%s;'
9          data = (USER_NAME,)
10         # Executing The Query From Function in utils file
11         res = executeSQLCommitQuery(query, data)
12         # Clearing Screen
13         clearScreen()
14
15         # If the result comes as a string it means a error has been raised e
16         lse Member Get Added
17         if(type(res) == str):
18             print("Error : ", res)
19         else:
20             print("Member Deleted")
21     except:
22         clearScreen()
23         print("Invalid Input")
```



```
1  def updateMemberPassword():
2      # Try Catch Block
3      try:
4          # Taking User Input
5          USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")
6
7          # Executing SQL Query
8          query = 'select * from members where username = %s;'
9          data = (USER_NAME,)
10         cnx = connection.MySQLConnection(
11             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
12             host='localhost', database=sqlData.DATABASE_NAME)
13         Cursor = cnx.cursor()
14         Cursor.execute(query, data)
15
16         # Creating PrettyTableObject
17         x = PrettyTable()
18         # Adding Field Names
19         x.field_names = ["User Name", "Date Of Registration"]
20
21         # Creating Found Variable
22         found = False
23         for i in Cursor:
24             # Storing Old Hashed Password to Compare
25             USER_PASSWORD = i[1]
26             # Using Datetime strftime to convert datetime object to string
27             dateString = date.strftime(i[2], "%d-%m-%Y")
28             data = [i[0], dateString]
29             x.add_row(data)
30             # Changing Found Variable TO True
31             found = True
32
33         # Printing Error If Not Found
34         if(not found):
35             x.add_row(["No User", "Record Found"])
36             print(x)
37             return
38
39         # Printing The Table
40         print(x)
41         # Closing the Connection
42         Cursor.close()
43         cnx.close()
```




```

1      # Getting User Input For Old Password
2      OLD_USER_PASSWORD = input(
3          f"{Colors.OKCYAN}Enter Your Old Password : {Colors.ENDC}")
4
5      # If Passwords Do not Match Raising an Exception
6      if(not checkPassword(OLD_USER_PASSWORD, USER_PASSWORD)):
7          clearScreen()
8          raise Exception(f"{Colors.FAIL}Wrong Password{Colors.ENDC}")
9
10     # Getting New Password Input From User
11     USER_PASSWORD = input(
12         f"{Colors.OKCYAN}Enter New User Password : {Colors.ENDC}")
13
14     # Checking If length of USER_PASSWORD is less than 4
15     # If raising an Exception
16     if(len(USER_PASSWORD) < 4):
17         raise Exception(f"{Colors.FAIL}Invalid Input{Colors.ENDC}")
18
19     # Hashing New User Password To Save it in sql table
20     USER_PASSWORD = hashPassword(USER_PASSWORD)
21
22     # Executing SQL Query
23     query = 'UPDATE members set password=%s where username=%s'
24     data = (USER_PASSWORD, USER_NAME)
25     # Executing The Query From Function in utils file
26     res = executeSQLCommitQuery(query, data)
27     # Clearing Screen For Better Presentation
28     clearScreen()
29     # If the result comes as a string it means a error has been raised else
30     Member Get Added
31     if(type(res) == str):
32         print("Error : ", res)
33     else:
34         print("Member Password Updated")
35 except Exception as err:
36     try:
37         print(handleSQLException(err))
38     except:
39         print(err)

```

issueReturn.py



```
1  # Importing ALL Functions From The utils file
2  from utils import *
3  # Importing datetime module
4  from datetime import date
5  # Importing PrettyTable from prettyprints module to print tables
6  from prettytable import PrettyTable
7  # Importing sqlData
8  import sqlData
9
10 # Setting The Fine per Day Variable
11 FINE_PER_DAY = 0.5 # 50 paise or 5 rupees
```



```
1  # Creating The Issue Book Function
2  def issueBook():
3      # Try Catch Block
4      try:
5          # Taking User Inputs
6          USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")
7
8          # Executing SQL Query
9          query = 'select * from members where username = %s;'
10         data = (USER_NAME,)
11         cnx = connection.MySQLConnection(
12             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
13             host='localhost', database=sqlData.DATABASE_NAME)
14         Cursor = cnx.cursor()
15         Cursor.execute(query, data)
16
17         # Creating Found Variable
18         found = False
19         for i in Cursor:
20             # Storing Hashed User Password To Compare it
21             HASHED_USER_PASSWORD = i[1]
22             # Changing Found Variable TO True
23             found = True
24
25         # If Not Found Raising Exception
26         if(not found):
27             raise Exception(f"{Colors.FAIL}User Not Found{Colors.ENDC}")
28
29         # Closing the Connection
30         Cursor.close()
31         cnx.close()
```



```
1      # Getting User Input For Password
2      USER_PASSWORD = input(f"{Colors.OKCYAN}Enter Password : {Colors.ENDC}")
3
4      # If Passwords Do not Match Raising an Exception
5      if(not checkPassword(USER_PASSWORD, HASHED_USER_PASSWORD)):
6          raise Exception(f"{Colors.FAIL}Wrong Password{Colors.ENDC}")
7
8      # Getting Input From User
9      BOOK_ISBN, DATE_OF_ISSUE = input(
10         f"{Colors.OKCYAN}Enter Book ISBN : {Colors.ENDC}"), date.today()
11
12     # Executing SQL Query
13     query = 'insert into issues Values(%s,%s,%s);'
14     data = (BOOK_ISBN, USER_NAME, DATE_OF_ISSUE)
15     # Executing The Query From Function in utils file
16     res = executeSQLCommitQuery(query, data)
17     # Clearing Screen For Better Presentation
18     clearScreen()
19     # If the result comes as a string it means a error has been raised else
20     # Member Get Added
21     if(type(res) == str):
22         print("Error : ", res)
23     else:
24         print("Issue Success")
25 except Exception as err:
26     clearScreen()
27     print(err)
```



```
1  def displayIssuedBooks():
2      # Try Catch Block
3      try:
4          # Executing SQL Query
5          query = "select * from issues;"
6          cnx = connection.MySQLConnection(
7              user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
8              host='localhost', database=sqlData.DATABASE_NAME)
9          Cursor = cnx.cursor()
10         Cursor.execute(query)
11
12         # Creating PrettyTableObject
13         x = PrettyTable()
14         # Adding Field Names
15         x.field_names = ["Book ISBN", "Username", "Date of Issue"]
16
17         for i in Cursor:
18             # Using Datetime strftime to convert datetime object to string
19             dateString = date.strftime(i[2], "%d-%m-%Y")
20             x.add_row([i[0], i[1], dateString])
21
22         # Printing The Table
23         print(x)
24
25         # Closing the Connection
26         Cursor.close()
27         cnx.close()
28
29     except Exception as err:
30         print(handleSQLException(err))
```



```
1  def returnIssuedBooks():
2      # Try Catch Block
3      try:
4          # Taking User Input
5          USER_NAME = input(f"{Colors.OKCYAN}Enter User Name : {Colors.ENDC}")
6
7          # Executing SQL Query
8          query = 'select * from issues where username = %s;'
9          data = (USER_NAME,)
10         cnx = connection.MySQLConnection(
11             user=sqlData.SQL_USERNAME, password=sqlData.SQL_PASSWORD,\
12             host='localhost', database=sqlData.DATABASE_NAME)
13         Cursor = cnx.cursor()
14         Cursor.execute(query, data)
15
16         # Creating PrettyTableObject
17         x = PrettyTable()
18         # Adding Field Names
19         x.field_names = ["Book ISBN", "User Name", "Date Of Issue"]
20
21         # Creating Found Variable
22         found = False
23         for i in Cursor:
24             # Storing Book Data
25             BOOK_ISBN = i[0]
26             DATE_ISSUED = i[2]
27             # Using Datetime strftime to convert datetime object to string
28             dateString = date.strftime(i[2], "%d-%m-%Y")
29             data = [i[0], i[1], dateString]
30             x.add_row(data)
31             # Changing Found Variable TO True
32             found = True
33
34         # Printing Error If Not Found
35         if(not found):
36             x.add_row(["No", "Issue Record", "Found"])
37             print(x)
38             return
39
40         # Printing The Table
41         print(x)
42         # Closing the Connection
43         Cursor.close()
44         cnx.close()
```



```

1      # Creating Days Issued Variable
2      DAYS_ISSUED = 0
3      # Calculating Days Issued
4      try:
5          DAYS_ISSUED = int(str(date.today() - DATE_ISSUED).split()[0]) - 7
6      except:
7          DATE_ISSUED = 0
8
9      # Changing Days To 0 if it is negative
10     if DAYS_ISSUED < 0:
11         DAYS_ISSUED = 0
12
13     # Getting User Input For Confirmation
14     choice = input(f"{Colors.WARNING} Do You Wish To Return The Book with ISBN {BOOK_ISBN} Associated With User {USER_NAME} The Fine Applicable Would Be Rs. {DAYS_ISSUED * FINE_PER_DAY}(y/n): ")
15
16     # Returning if choice is not y/Y
17     if(choice.lower() != "y"):
18         clearScreen()
19         return
20
21     # Executing SQL Query
22     query = 'delete from issues where username=%s'
23     data = (USER_NAME,)
24     # Executing The Query From Function in utils file
25     res = executeSQLCommitQuery(query, data)
26     # Clearing Screen For Better Presentation
27     clearScreen()
28     # If the result comes as a string it means a error has been raised else
29     # Member Get Added
30     if(type(res) == str):
31         print("Error : ", res)
32     else:
33         print("Issued Record Deleted")
34 except Exception as err:
35     clearScreen()
36     print(err)

```




```
1
2  def menu():
3      # Creating The Issue/Return Options Dictionary
4      issueReturnBookMenuOptions = {
5          1: ["Issue Book", issueBook],
6          2: ["Display Issued Books", displayIssuedBooks],
7          3: ["Return Issued Books", returnIssuedBooks],
8          4: ["Return To Main Menu"]
9      }
10     # Running The Show Menu Function From Utils File On The Main Menu Options Dictionary
11     # Passing Members Menu as Menu Title and Menu Spacing from utils file to center the menu heading
12     showMenu("Issue Return Menu", issueReturnBookMenuOptions, menuSpacing)
```

Output

MAIN MENU

=====

Main Menu

=====

- 1 - Book Management
- 2 - Members Management
- 3 - Issue / Return Book
- 4 - Exit

Enter Choice :

BOOKS MENU

Books Menu

=====

- 1 - Add Book
- 2 - Display Books
- 3 - Delete Book
- 4 - Search Book
- 5 - Update Book
- 6 - Return To Main Menu

Enter Choice :

ADD BOOK

Enter Book Name : Manas Madan

Enter Book ISBN : 1234

Enter Book Author : Manas

Enter Book Publishing Date : 12

Enter Book Publishing Month : 10

Enter Book Publishing Year : 2020

DISPLAY BOOKS

Book Name	Book ISBN	Book Author	Date Of Publishing
Manas Madan	1234	Manas	12-10-2020

SEARCH BOOKS

Enter Book ISBN : 1234

Book Name	Book ISBN	Book Author	Date Of Publishing
Manas Madan	1234	Manas	12-10-2020

Enter Book ISBN : 123

Book Name	Book ISBN	Book Author	Date Of Publishing
No	Book	Record	Found

UPDATE BOOKS

Enter Book ISBN : 1234

Book Name	Book ISBN	Book Author	Date Of Publishing
Manas Madan	1234	Manas	12-10-2020

Enter Book Name : Manas

Enter Book Author : Manas Madan

Enter Book Publishing Date : 10

Enter Book Publishing Month : 12

Enter Book Publishing Year : 2020

Enter Book ISBN : 123

Book Name	Book ISBN	Book Author	Date Of Publishing
No	Book	Record	Found

DELETE BOOK

Enter Book ISBN : 1234

MEMBERS MENU

Members Menu

- ```
=====
```
- 1 - Add Member
  - 2 - Display Members
  - 3 - Search Members
  - 4 - Delete Member
  - 5 - Update Member Password
  - 6 - Return To Main Menu
- Enter Choice :

## ADD MEMBER

Enter User Name : Manas

Enter User Password : password

```
mysql> SELECT * FROM MEMBERS
-> ;
```

| username | password                                                        | Date_of_Registration |
|----------|-----------------------------------------------------------------|----------------------|
| Manas    | \$2b\$12\$RMSDPZS93c6.kN3YmeBt6.Ud1duqb5.N2JLxI5KNX23McaM5uFxGC | 2021-12-26           |

## DISPLAY MEMBERS

| User Name | Date Of Registation |
|-----------|---------------------|
| Manas     | 26-12-2021          |

## SEARCH MEMBERS

Enter User Name : m

|               |                      |
|---------------|----------------------|
| +-----+-----+ |                      |
| User Name     | Date of Registration |
| +-----+-----+ |                      |
| No User       | Record Found         |
| +-----+-----+ |                      |

Enter User Name : Manas

|               |                      |
|---------------|----------------------|
| +-----+-----+ |                      |
| User Name     | Date of Registration |
| +-----+-----+ |                      |
| Manas         | 26-12-2021           |
| +-----+-----+ |                      |

## DELETE MEMBER

Enter User Name : Manas

## UPDATE MEMBER PASSWORD

Enter User Name : Manas Madan

|               |                      |
|---------------|----------------------|
| +-----+-----+ |                      |
| User Name     | Date Of Registration |
| +-----+-----+ |                      |
| Manas Madan   | 26-12-2021           |
| +-----+-----+ |                      |

Enter Your Old Password : password

Enter New User Password : newpassword

# ISSUE MENU

## Issue Return Menu

=====

- 1 - Issue Book
- 2 - Display Issued Books
- 3 - Return Issued Books
- 4 - Return To Main Menu

Enter Choice :

# ISSUE BOOK

Enter User Name : Manas Madan

Enter Password : newpassword

Enter Book ISBN : 1234

# DISPLAY ISSUED BOOKS

|                     |             |               |
|---------------------|-------------|---------------|
| +-----+-----+-----+ |             |               |
| Book ISBN           | Username    | Date of Issue |
| +-----+-----+-----+ |             |               |
| 1234                | Manas Madan | 26-12-2021    |
| +-----+-----+-----+ |             |               |



# RETURN ISSUED BOOK

Enter User Name : Manas Madan

|                     |             |               |
|---------------------|-------------|---------------|
| +-----+-----+-----+ |             |               |
| Book ISBN           | User Name   | Date Of Issue |
| +-----+-----+-----+ |             |               |
| 1234                | Manas Madan | 26-12-2021    |
| +-----+-----+-----+ |             |               |

Do You Wish To Return The Book with ISBN 1234 Associated With User Manas Madan The Fine Applicable Would Be Rs. 0.0(y/n): y

---

## Modules Used

- bcrypt - 3.2.0
  - To Hash Passwords
- mysql-connector-python - 8.0.27
  - To Establish Connection between python code and mysql database
- prettytable - 2.5.0
  - To print sql tables beautifully
- datetime - inbuilt
  - To add in DATE column of sql

## BIBLIOGRAPHY & REFERENCES

- [SQL Reference](#)
- [Colored Python Terminal](#)
- [Python and MySQL Connector Reference](#)
- [MySQL Python Connector Docs](#)