PROJECT E-libraries

Hy359 .Δάβανος Ιωάννης csd4622/ Ηλίας Καψής csd4652

INTRO

This website was developed with the aim to address the needs of students and librarians in the field of book and research. The site was designed HTML ,CSS ,JAVASCRIPT with a user friendly interface and simple UI. It allows the students to search for books available in libraries and request them for a loan for a specific time period. Librarians , can upload books and give permission to students to loan books. The books, student and librarian accounts are stored in a mysql database. The mysql database contains 7 tables which are : Adminmessages , Books, BooksInLibraries,Borrowing,Reviews,Librariran,Students. The communication betweens server and clients is handled by Java Servlets with php commands , and Rest Api. Javascript is used for calling servlets and handling the users requests accordingly.

HTML files

- ***BookSearch.html***
  - This is a basic HTML code for a webpage to search books. The page contains a form with various fields for input such as title, author, minimum and maximum number of pages, genre, and publishing year range.
  - The form submits the data through a JavaScript function "SearchforBook()" which is triggered by a button labeled "Search".
  - The result of the search is displayed in a div with the id "ajaxContent"
- ***NewBookPage.html***
  - This is a HTML code for a form to add a new book. The form contains fields for inputting book details such as ISBN, title, author, genre, number of pages, publication year, URL and a photo.
  - The form has a submit button that triggers the JavaScript function "AddNewBook()". The form data is not submitted to the server via a traditional HTTP request, as the form's "onsubmit" event handler returns false, preventing the default form submission behavior.
- ***ReviewPage.html***
  - The title of the page: "Pie Options"
  - The encoding: "UTF-8"
  - The compatibility mode: "IE=edge"
  - The viewport meta tag for device-width and initial-scale
  - A link to a stylesheet file: "./resources/css/sign-up.css"
  - A Google Charts API script: "https://www.gstatic.com/charts/loader.js"

    - An AJAX script file: "js/ajax.js"
    - A jQuery script: "https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
    - A body section with:
    - A page div with a header "Reviews" and two buttons: Logout and Back
    - An AJAX content div with an id "ajaxContent"
    - A footer section with a list of links and a copyright notice.
    - A div container with an id "container".
- ***SIgnUp.html***
    - ***This html is used to register a new user***
- ***Admin.html***
    - ***This html is the main page of the admin***
- ***Books.html***
    - ***A table of all the books***
- ***Index.html***
    - ***The main page***
- ***Libchoices.html***
    - ***The choices that are aailable for the librarian user***
- ***Login.html***
    - ***The part of html that shows the login for the user***
- ***Loginoptions.html***
    - ***The options for the student user***
- ***Options.html***
    - ***The page where users can change their information***
- ***Statistics.html***
    - ***The part of the page where chart pies are shown***

We also have a css file to make our front end look nice

JavaScript functions

- ***function UserPOST(): It checks the user type and calls the appropriate function to place the user to the correct table. Calls registerStudent servlet***
- ***function StudentChecker() adds student to the Object Student in database.***
- ***function LibrarianChecker() adds Librarian to the Object Librarian in database.Calls RegisterLibrarian servlet***
- ***function NameCheck() checks if the username the user user provides is appropriate. Calls UsernameChecker servlet .***
- ***function EmailCheck() checks if the user provided an email account . Calls email checker servlet***
- ***function PassCheck() checks if the number of the pass exists , if it doesn't it displayes an error. Calls StudentIdChecker***
- ***buildHtmlTable(myList) creates a table***
- ***function addAllColumnHeaders(myList) is used by buildHtmlTable() to build the table***
- ***function showLogin() show the login page to user and allow him to enter the site with his credentials .Calls Login Servlet***
- ***function showRegistrationForm() shows the registration page to user and allows him to create an account for the page and access its features .***
- ***function showUserMainPage() Takes user to the sign up page***
- ***function loginPOST() If user is exists it logs him in and it gives him access to the site's abilities.Calls Login***
- ***function settingPOST() updates user credentials after user does so.Calls Settings servlet***
- ***function showSettings() gives the ability to user to change his credentials***

- ***function logout() Logs the user out and takes him to starting page . Calls Logout servlet***
- ***function CheckOnLoad() is being called each time to check if the window ends to option html . Calls Settings Servlet***
- ***function showBooks() shows each unique Book that exist in EveryLibrary .Calls Books servlet***
- ***function showUsers() can be accessed only by admin and it shows him all the students that have created accounts.Calls Users servlet***
- ***function deleteUser(usr) only admins can access it and allows gives him the option to delete user and his credentials from database.So even if the user tries to log back in he wont be able to , because his account no longer exists. Calls DeleteUser servlet***
- ***function back() it's rarely used only for taking the user to the previous page***
- ***function showStatisticks() takes admin to the statistics page where he can see pie charts about information in the database .***
- ***function ShowBooksPerLibrary() is a statistic that shows how many books each Library has in their collection***
- ***function drawPieChart1() draws admin the pie chart for ShowBooksPerLibrary. Calls server StatisticBookPerLibrary***
- ***function ShowBooksPerGenre() Shows how many books are per Genre***
- ***function drawPieChart2() draws admin the pie chart for ShowBooksPerGenre(). Calls servlet StatisticBookPerGenre***
- ***function NumberPerLevel() Shows how many students are at each Academic level***
- ***function drawPieChart3() draws the piechart for NumberPerLevel(). Calls StatisticStudentsPerLevel servlet***

- ***function searchBooks() gives student the ability to search books with specific filters***
- ***function SearchforBook() after the students searches for bookis with specific filters it prints him the relevant results .Calls BooksSearch servlet***
- ***function LoanBook(data) After the user Decides to loan a book , it creates a request and a servlet is called where it uploads his request . Calls LoanBook servlet***
- ***function showLib() it is called when the student want to know whats the quickest route to the nearest library compared to his location . The user will be prompted by the browser to ask for permission of his location . Calls Location servlet***
- ***function showQuickestRoute(UserLat, UserLng, LibraryLat, LibraryLng) Shows a Google map with the quickest route to closest library. To do that the user provides his coordinates and I take the closest Library;s coordinates.***
- ***function CheckforLoan() it is called if the user wants to see any incoming notification. A notification is shown if a book the user has loaned is due to for a return in 3 or less days .Calls Notify servlet***
- ***function Reviews(str) when the user presses the review button from a book , it will show him all reviews of that specific book.Calls GetRrviews servlet***
- ***function ReviewBook(data) if the user presses the Review button this function is called and allows the user to give a review for the specific book with isbn data.Calls GetReviews sevlet***
- ***function Inventory() when button inventory is pressed it shows the user all the books he has ever loaned and gives option to return a book he hasn't returned yet . Calls inventory servlet***

- ***function Return(data) returns the book the user chose. Calls Return servlet .***

REST

In the rest api we have two post methods that take a json file and return a json file

The methods are:

- ***AddBook()***
  - ***Its path : http://localhost:8080/mavenproject1/library/books/newBook***
  - ***The service first converts the JSON string into a Book object using the jsonToBook() method from the EditBooksTable class. Then it checks if the book already exists in the database using the checkBook() method, and returns a 409 Conflict status if it does. If the book is new, it is added to the database using the addBookFromJSON() method and a 200 OK status is returned to indicate success.***
- ***AddLibraryBook()***
  - ***Its path: http://localhost:8080/mavenproject1/library/books/availableBook***
  - ***The service first converts the JSON string into a BookInLibrary object using the jsonTobookInLibrary() method from the EditBooksInLibraryTable class. Then it checks if the book is already in the library using the Checker() method and if it exists, it updates the availability of the book using the updateBookInLibrary() method. If the book is new and exists in the books table, it is added to the library using the addBookInLibraryFromJSON() method. If***

the book is neither new nor exists in the books table, a 409 Conflict status is returned to indicate an error.

MainClasses Packet

- ***AdminMessage(int message_id,String message,date)***
- ***Book(String isbn, title, authors, genre, url, photo, int pages, publicationyear)***
- ***BookInLibrary(String isbn,*** int bookcopy_id,library_id,String available)
- ***Borrowing(int borrowing_id,bookcopy_id,user_id, String fromDate,toDate,status)***
- ***JSON_Converter is used to convert BufferedReader into String***
- ***Librarian(int library_id, String libraryname,libraryinfo)***
- ***Library(int id, int books)***
- ***Review(int review_id,user_id, String isbn,reviewText,reviewScore)***
- ***Student(int user_id, String student_id, university,department,student_type, student_id_from_date,student_id_to_date)***
- ***User(String username,email,password,firstname,lastname,birthdate, gender, country,city,address, lat,lon, telephone, personalpage)***

Every class that has properties has setters and getters for them.Also, student and librarian classes are subclasses of User

Database.tables packet

*EditAdminMessages: **provides functionality to manage a database table named "adminmessages". It can perform operations such as adding a new admin message to the database, converting a JSON representation of an AdminMessage object to an AdminMessage object and vice-versa, and retrieving an AdminMessage from the database based on its ID. The code uses the Gson library to handle the JSON conversions. The code also uses the DB_Connection class to handle database connections and queries.***

*EditBooksInLibrary: **The class "EditBooksInLibraryTable" provides methods for adding, transforming, and retrieving information related to the "booksinlibraries" database table. It has methods for converting a BookInLibrary object to and from JSON, as well as methods for retrieving BookInLibrary objects from the database using either the book copy ID or the ISBN and library ID. Additionally, there is a method for creating the booksinlibraries table in the database.***

*EditBooksTable: **This class contains several methods for adding, updating, and retrieving book data from a database. The class uses Google's GSON library to convert between JSON strings and Book objects, which represent individual books. The class connects to a database using a database connection class (DB_Connection).***

The methods in this class allow for adding new books to the database, converting between JSON strings and Book objects, retrieving all books from the database, checking if a book is in the database, retrieving books from the database based on a specified genre,

updating books in the database, deleting books from the database, and creating the books table in the database if it does not already exist.

EditBorrowingTable: ***This class is part of a library management system that manages borrowing books. It can perform operations like adding a new borrowing record, updating the status of a borrowing record, deleting a borrowing record, and creating the borrowing table in the database. The class uses the DB_Connection class to connect to the database and perform CRUD operations. The class uses the Borrowing class to store the borrowing data and performs operations like converting Borrowing data to/from JSON using the Gson library.***

EditLibrarianTable: ***This is a Java class for updating librarians in a database. It provides methods to update librarian personal information, such as name, birthday, gender, and location, as well as methods to convert a librarian object to and from JSON representation. The class uses the GSON library to handle the JSON conversion and the JDBC API to connect to the database and execute SQL statements.It's worth noting that the code uses string concatenation to build the SQL statements, which is susceptible to SQL injection attacks. Instead, consider using prepared statements or stored procedures to execute the updates.***

EditReviewsTable:

This is Java code for the class "EditReviewsTable". It provides methods for adding, retrieving, and manipulating book review

information in a database. The methods include adding a review from a JSON string, converting a review to and from a JSON string, retrieving a review from the database using its ID, retrieving multiple reviews from the database using the ISBN of a book, creating the review table in the database, and creating a new review entry in the database.

EditStudentsTable:

The class EditStudentsTable provides a number of methods to add, update and retrieve information from a database table that stores information about students.

addStudentFromJSON: Adds a new student to the table, by converting the JSON string of student information to a Student object, and calling the addNewStudent method to persist it in the database.

jsonToStudent: Converts a JSON string to a Student object using the GSON library.

studentToJSON: Converts a Student object to a JSON string using the GSON library.

updateStudent: Updates a student's information in the database table by executing a SQL update statement with the new information provided. There are similar methods to update other student information such as password, first name, last name, birthdate, gender, country, city, address, and email.

GeneralQueries:

The GeneralQueries class provides methods to fetch data from the database and return them as ArrayList or JsonArray objects.

The allLibrariesHavingABookAvailable method returns a list of librarians of libraries having a book with a given ISBN number.

The allBooksOfALibrary method returns the details of all books present in a library with a given library id.

The allBorrowingsOfALibrary method returns the details of all book borrowings from a library with a given library id.

Servlets Packet

***Books**: This servlet handles HTTP GET requests for displaying information about books. The servlet connects to a database table named EditBooksTable and retrieves data for books to create a table of books. The data for each book is stored as a Book object, which is part of the mainClasses package. The servlet generates a table and sends it as a response in HTML format to the client. The doPost method receives an HTTP request and response as input.It first converts an incoming JSON payload from the request into a Book object using the jsonToBook method from the EditBooksTable class.Then it checks if the book with the given ISBN number already exists in the database using the checkBook method of the EditBooksTable class.If the book with the given ISBN number exists, it sets the status code of the HTTP response to 409 (Conflict).Otherwise, it sets the status code of the HTTP response to 200 (OK).Finally, it sets the content type and character encoding of the HTTP response to "application/json" and "UTF-8" respectively.*

***BookSearch:** This is a Java Servlet that handles a HTTP POST request to search for books in a database based on certain criteria, such as title, genre, author, publication year and number of pages. The servlet receives the search criteria in a JSON format from an AJAX request. The search criteria are extracted and used to filter books from the database. The filtered books are then returned in an HTML table format.*

*ChangeStatus: **This code implements a Java servlet (ChangeStatus) which handles a HTTP POST request. When this servlet is called, it does the following actions:***

1.Creates instances of two classes EditBorrowingTable and EditBooksInLibraryTable.

2.Parses the values of two request parameters "intValue" and "stringParam" from the HTTP request.

3.If the value of "stringParam" is "successEnd", updates the "book in library" table by setting the value of the "available" field to "true".

4.Calls the updateBorrowing method of the EditBorrowingTable class and passes the values of "id" and "status" as arguments, which updates the borrowing table with the new status.

5.Logs any SQLException or ClassNotFoundException caught during the execution of the servlet.

DeleteUser:

1.Creates an instance of the JSON_Converter class and a PrintWriter for writing the response.

2.Sets the content type and character encoding of the response to "application/json" and "UTF-8", respectively.

3.Reads the data from the request's reader and stores it in the "data" variable.

4.Uses the Gson library to convert the data from a JSON string to a User object.

5.Gets the username of the User object and creates an instance of the EditStudentsTable class.

6.Tries to remove the student using the "RemoveStudent" method and sets the status of the response to 200 on success.

7.Catches any SQLException or ClassNotFoundException and sets the status of the response to 501 or 504, respectively.

EmailChecker: ***This code implements a Java servlet (EmailChecker) which handles a HTTP POST request. When this servlet is called, it does the following actions:***

1. The method retrieves a JSON string representation of a "Student" object from an HTTP request using the getJSONFromAjax method of the "JSON_Converter" object.

2. The method then converts the JSON string to a "Student" object using the "jsonToStudent" method of the "EditStudentsTable" object.

3. The method then uses JDBC to connect to a database and execute two SELECT statements to check if the email address of the "Student" object already exists in either the "students" table or the "librarians" table.

4. If it does, the method sets the HTTP response status to 403 (Forbidden). Otherwise, it logs the JSON string representation of the "Student" object to the console.

GetReviews: ***This Java code is a servlet method (doPost) handling a POST request in Java Servlet GetReviews. This code when called***

1. The method sets the response content type to "application/json" and character encoding to "UTF-8".

2. It retrieves the data from the request body using a BufferedReader, converts it to a string and saves it in a variable "data".

3. The method then creates instances of the "EditReviewsTable" and "EditBooksTable" classes.

4. Using the "databaseToReviews" method of the "EditReviewsTable" object, the method retrieves an ArrayList of "Review" objects from the database, filtered by the "data" value.

5. If the ArrayList is not null, the method iterates through the list to extract information about a "Book" object, creates an HTML string and appends it with review information, and the ISBN number of the book.

6. The method returns the HTML string in the response body.

Inventory: *This Java Servlet (Inventory) code generates an HTML response to display a table of books borrowed by a logged in user.*

1. The code first gets the session object with HttpSession session = request.getSession(true); and retrieves the username attribute from the session with String sr = (String) session.getAttribute("username");

2. It then uses the EditStudentsTable class to retrieve the student data based on the username.

3. The EditBorrowingTable class is used to get a list of borrowing records for the student, and the EditBooksInLibraryTable and

EditBooksTable classes are used to retrieve information about the books.

4. The code builds the table's rows by looping over the list of borrowing records and adding each book's information to the table.

5. If a book is successfully borrowed, a "Return" button is added to the row. The HTML is written to the response with out.println(mystring)

LoanBook: This is a Java Servlet code (LoanBook) for handling a POST request in a web application. The code performs the following operations:

1. Get the print writer from the response object to output the result of the operation.

2. Set the content type and character encoding of the response to "application/json" and "UTF-8" respectively.

3. Read the data sent in the request body and store it in the data variable.

4. Get the session associated with the request. If the session exists, proceed to the next step, else output "error - Please Sign in".

5. Get the Book object by calling the getBookswithId method of the

Location: This Java code is a Servlet (Location)for handling a post request and returning a JSON object of the nearest librarian to a given latitude and longitude.

1. It starts by creating a JSON converter and print writer, setting the content type and character encoding of the response.

2. Then, it reads the data sent in the request and uses the Gson library to parse the JSON into a Book object. The latitude and longitude are extracted from the Book object.

3. The code then retrieves a list of all librarians from a database using the EditLibrarianTable class and calculates the distance from each librarian to the given latitude and longitude.

4. The librarian with the shortest distance is selected and returned as a JSON object in the response.

5. The response status is set to 200 to indicate success.

LoggedChecker: ***This code is the implementation of the doGet method of a Servlet.***

1. The method retrieves the current HTTP session and retrieves the attributes "username" and "password" from it.

2. Then it checks if the values of "username" and "password" are "admin" and "admin12*" respectively.

3. If the values match, it sets the HTTP status code of the response to 200 (OK).

4. If the values don't match, it sets the HTTP status code of the response to 403 (Forbidden).

Login: ***This code is a Java Servlet that implements the HttpServlet class and overrides the doGet and doPost methods***

The doGet method handles the HTTP GET request and retrieves the username and password from the session

1. if the username and password match "admin" and "admin12*", respectively, it returns status code 200, otherwise it returns status code 403.

The doPost method handles the HTTP POST request and reads a JSON object from the request body, which contains a user's username and password.

1. It then uses the Gson library to deserialize the JSON string into a User object.

2. It checks if the user is a student, librarian, or an admin, and sets the session attributes accordingly.

3. If the user is not found in the database, it returns status code 401.

Logout: ***This code implements the HTTP GET and POST methods for a servlet.***

The GET method is used to log out a user and invalidate their session, which is indicated by removing the "username" and "password" attributes from the session object.

The POST method is used to handle a user's login attempt.

1. If a session with the username attribute does not exist, a status of 403 is returned and the "index.html" page is displayed.

2. if a session with the username attribute exists, a status of 200 is returned and the "loginchoices.html" page is displayed.

Notify: ***This code is a Java servlet(Notify) that implements the doGet method, which processes a GET request. When this servlet is called, it does the following:***

1.Creates a PrintWriter object out to write to the response.

2.Sets the content type and character encoding of the response to "application/json" and "UTF-8" respectively.

3.Gets the current session using the request.getSession(true) method.

4.Gets the value of the "username" attribute from the session and stores it in a string sr.

5.Creates an instance of the EditStudentsTable class and uses it to retrieve the student data associated with the "username" attribute.

6.Creates an instance of the EditBorrowingTable class and uses it to retrieve books that are due to be returned by the student.

7.If there are no books due to be returned, the servlet writes "No notifications" to the response. If there are books due to be returned, it writes a string to the response that lists the books that are due to be returned.

8.Flushes the output stream.

RegisterStudent: ***This Java code is a servlet that implements the doPost method. When the servlet is accessed via a HTTP POST request, this method will be executed.***

1. It converts the request body into a Student object by using a custom class called "JSON_Converter" and checking if the student's username and email are not already taken in the database using the methods "databaseStudentChecker" and "databaseToLibrarianChecker".

2. If the username and email are unique, the student data is stored in the database by calling the "addStudentFromJSON" method.

3. If the username or email are already taken, a JSON error message is sent back to the client with a status code of 409 (Conflict).

RegisterLibrarian: ***This Java code is a servlet that implements the doPost method. When the servlet is accessed via a HTTP POST request, this method will be executed.***

1. It converts the request body into a Librarian object using the "jsonToLibrarian" method from the "EditLibrarianTable" class.

2. It then checks if the Librarian's username and email are not already taken in the database using the methods "databaseStudentCheckerL" and "databaseToLibrarianChecker".

3. If the username and email are unique, the Librarian data is stored in the database by calling the "addLibrarianFromJSON" method.

4. If the username or email are already taken, a JSON error message is sent back to the client with a status code of 403 (Forbidden).

Return: ***This Java code is a servlet that implements the doPost method. When the servlet is accessed via a HTTP POST request, this method will be executed.***

1.Create a JSON_Converter object, set the response content type to "application/json", and set the character encoding to "UTF-8".

2.Read the request body as a string, store it in "data".

3.Create objects for EditBooksInLibraryTable and EditBooksTable, which are classes for managing books in the library and books.

4.Use the EditBooksInLibraryTable object to check if a book with the specified ISBN exists in the library. If it exists, store it in the "ok" object.

5.Use the EditBooksTable object to get a book with the specified ISBN. If it exists, store it in the "po" object.

6.Use the EditBooksInLibraryTable object to update the loan status of the book to "true".

7.Create an object for EditBorrowingTable, which is a class for managing borrowing data.

8.Use the EditBorrowingTable object to get borrowing information for the book copy associated with the ISBN.

9.If there are exceptions, log them.

Settings: This code is a Servlet class in Java, which handles HTTP requests made to the server. It has two methods, doGet and doPost

DoGet: retrieves the user profile information and returns it to the client as a string.

DoPost:1. updates the user profile information in the database, by converting a JSON string from the client into a student object, using the JSON_Converter class.

2. The profile update is done for either a student or librarian, depending on who is currently logged in, by using the EditStudentsTable or EditLibrarianTable classes.

3. The response status code is set to 200 for a successful update and 401 for a failure.

StatisticPerBookGenre: ***This is a servlet method in Java using the Servlet API. It calculates the statistics of books per genre***

1.First, creating an instance of EditBooksTable class, which is likely a class that interacts with a database to retrieve information about books.

2. The method then retrieves all books using the getAllBooks() method of the table instance and adds each unique genre to an arraylist called "array".

3. It then creates another arraylist "ina" which holds the count of books for each genre by calling the getBookswithGenre(array.get(i)) method.

4. Finally, it converts the result arraylist to a JSON representation using GSON library and sends the result back to the client in the response body with a content-type of "application/json" and sets the status code of the response to 200 (OK).

StatisticBookPerLibrary: ***This code is a Java servlet, which processes GET requests and retrieves information from a database. It retrieves information about books in the library and the number of books available in each library and returns the information as a JSON response.***

1. The servlet uses the EditBooksInLibraryTable class to connect to the database and retrieve the information

2. If there is any SQL or ClassNotFound exception, it sets the response status to 500 and logs the exception.

3. If the processing is successful, it sets the response content type to "application/json" and writes the information as a JSON string to the response using the Gson library.

StatisticStudentsPerLevel: ***This code is a servlet for handling HTTP GET requests. It retrieves information about students from a database and sends it to the client as a JSON response. The servlet does the following steps:***

1.Creates an instance of the EditStudentsTable class, which is presumably a class for interacting with a database table of student information.

2.Initializes variables under, post, and bachel to keep track of the number of students in each category (undergraduate, graduate, and masters, respectively).

3.Retrieves a list of students from the database using databaseToStudents() method of the EditStudentsTable instance.

4.Iterates through the list of students, counting the number of students in each category.

5.Clears the list of students.

6.Creates three instances of the Student class, each representing one category of student (undergraduate, graduate, and masters).

7.Adds the three student instances to the list of students.

8.Serializes the list of students to JSON using the GSON library.

9.Sends the JSON response to the client.

10.Sets the HTTP response status code to 200 if the operation is successful or to 500 if there is an exception thrown (e.g. due to a database error).

StudentIdChecker: ***This Java servlet receives an HTTP POST request and processes a JSON string contained in the request body.***

1. It uses the class JSON_Converter to parse the JSON string, then creates a Student object from the parsed data.

2. The servlet checks if the student ID already exists in the students table of the database.

3. If it does, the servlet sends a response with a status code of 403 (Forbidden).

4. If the student ID does not exist, the servlet sends a response with a status code of 200 (OK).

UsernameChecker: ***This Java servlet receives an HTTP POST request and processes a JSON string contained in the request body.***

1. It uses the class JSON_Converter to parse the JSON string, then creates a Student object from the parsed data.

2. The servlet checks if the student ID already exists in the students table of the database.

3. If it does, the servlet sends a response with a status code of 403 (Forbidden).

4.If the student ID does not exist, the servlet sends a response with a status code of 200 (OK).

<u>Users: This is a Java Servlet (Users)that serves a GET request.</u>

<u>1. When this servlet is called, it retrieves information about students from a database using the EditStudentsTable class, then formats the data as an HTML table to be returned as the response to the GET request.</u>

<u>2.</u> The response is sent as UTF-8 encoding, and it sets a HTTP status code of 200 if the data retrieval and formatting are successful, and a status code of 500 if there are any exceptions, such as a failure to access the database or a failure to encode the response.

Admin Fuctionality

In this section we will briefly explain , what we have implemented for the admin. All of admin's requested capabilities have been implemented. Firstly, the admin logs in by entering the credentials username: admin password:admin12*. After logging in, the admin has two option .

1. ***See the users that have created an account , and he has the ability to delete their accounts.***
2. ***See some statistics about the database.***

Student Fuctionality

In this section we will briefly explain , what we have implemented for Student.The student has the ability to change his credentials after logging in. He has the ability to see all the books and their relative information. He can search for Books using specific filters to find books with the user's preferences. The student can also loan a book for one month if the book he is trying to loan is available and at least one copy is available from one library . The students also has a notification button , where he can see if he has any books due to for a return . We have implemented some code to allow the user to see reviews for a book, but it has a bug we havent detected and it doesn't show the reviews on page . We also have implemented the code where it allows the user to see all the books that he has ever gotten a loan for.The one bug it has is, we coded a return button with a function that returns a book to the server but for some reason the button is not shown on screen. Moreover, the student has the ability to share his location and the site will show him the Quickest route for the Nearest Library. The sevlet that calculates the coordinates has been coded, but there is a bug where our website cant import the google maps api(don't know why) and the map doesn't show , even though we have coded it to appear.

Librarian Fuctionality

In this section we will briefly explain , what we have implemented for Librarian.First of all , the librarian can change his credentials if wanted to .The librarian has the ability to add info to books where there isnt any.He can also add new books. Both of these were implemented using Rest Api.The librarian can also check if the library is available from certain libraries.The librarian can check for requested book and loan them out to the user that asked for it . There is bug on the part where the librarian checks all the books that have been loaned out by a specific library.And the pdf that the librarian can download only shows bookcopyid.

Guest Fuctionality

In this section we will briefly explain , what we have implemented for Guest. The Guest capabilities are limited since he is not registered on the website. He can do all the tasks provided by the Assignment . He can see all of the books and all the relative links about the books and other univestities of Crete.