

DEVSHelf - ERROR 204

1. Learning Outcomes

- Started work initially with just HTML and CSS but as time progressed we encountered Bootstrap where we came across many useful features that we used in our website to enhance the user experience.
- Learned how the backend of websites worked through the help of functions like GET and POST and implemented these with the help of Node.js and Express.
- Understood the functionalities of databases while using MongoDB and also its importance in managing large data systems like those of a library. MongoDB turned out to be very useful to us considering its capabilities of creating different collections under a single database as this helped us in including many additional features.
- We also came across many functions that we used in our website to enhance the features of the websites to increase user satisfaction. Some of these functions were nodemailer and node-cron.
- Nodemailer was used in one of our best features, the OTP generation and verification during registration.
- Node-cron was used to manage dates and times which was very helpful for implementing our reminder feature that would let the user know they had to return a book once they had reached the due date.

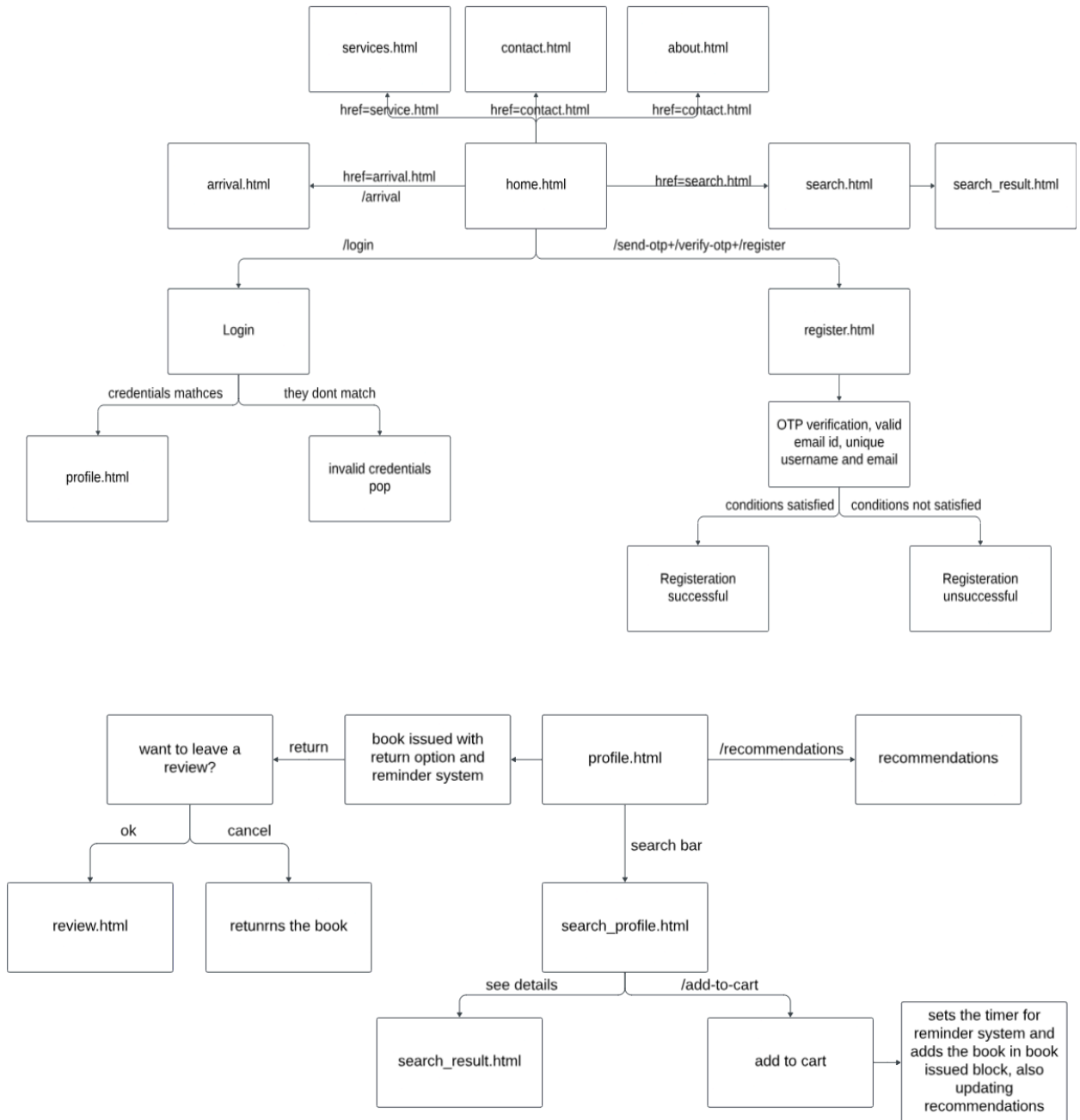
2. Challenges Faced

- One of our major challenges we faced was the inclusion of the OTP feature during registration. We were able to generate an OTP, send it to the user's email ID and also verify it, but when we were required to fetch and view the user's profile details, we were having problems. It was noted that the OTP generation was creating a second document in the collection for the same user in addition to the first one which contained all the details of the user. Nothing was being shown as the required tokens hadn't been declared in the 2nd document. To resolve this issue, we modified our code such that the second

document was not only being deleted as soon as the user was registered but also stored in another collection for record purposes.

- To implement OTP verification for user registration, we used Nodemailer to send emails. Initially, we believed the password required for Nodemailer was our Google account password, but we discovered that we needed to generate an app-specific password. We enabled two-step verification on our Google account and generated an app-specific password for Nodemailer. Additionally, we ensured our Google account settings allowed third-party applications to send emails via API. This setup allowed us to securely send OTPs to users for email verification during registration. Before using Nodemailer, we considered SMTP but encountered limitations, including the ability to send emails only to our own account without a premium subscription. These constraints led us to choose Nodemailer, which provided the necessary functionality without requiring a premium subscription.
- We tried to create an automatic book returning system once the due date of the book issued approached. We used the node-cron function to take note of the date and time of the book being issued and worked on making the book get automatically returned after a given number of days. Although we tried to work on this and after several unsuccessful attempts decided to drop it because we realised that the automatic returning of books would have to require admin administration and hence decided to go ahead with a method that sent reminder emails to the user asking them to return the book.
- Another challenge we faced was ensuring that there were no users in our database with duplicate usernames and email IDs. This means that once a user registered with a particular username and email ID, no other person could use either of them while registering. Although we had some problems in executing this initially, at the end we were able to make some changes in our code and ensure this feature worked.
- Lastly, although not a major challenge, we did have issues with the installation of MongoDB. But nevertheless they were sorted out in no time without interrupting the rhythm of our work.

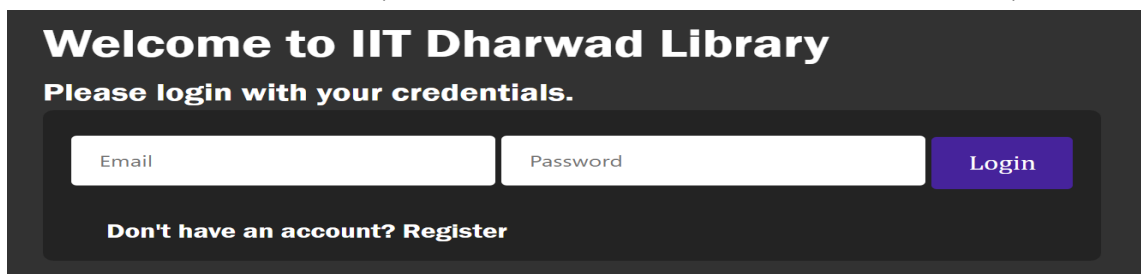
3.Flowchart of our website



4.Implemented Features and their Details

A. Compulsory Features

- To begin with, we have the home page that navigates to all other pages of the website showing us all the features the library has to offer.
- We have a login/register section that enables new users to create an account as well as the previous users to login and view their profile.



The image shows a login form for the IIT Dharwad Library. It has a dark background with white text. At the top, it says "Welcome to IIT Dharwad Library" in a large, bold font. Below that, it says "Please login with your credentials." in a smaller font. There are two input fields: one for "Email" and one for "Password". To the right of the "Password" field is a blue button labeled "Login". Below the input fields, there is a link that says "Don't have an account? Register".

Welcome to IIT Dharwad Library

Please login with your credentials.


Email Password Login

Don't have an account? Register

- The register link leads to a new page where the user is asked for their credentials like username, email ID, password, favourite book and favourite author. An OTP is generated which must be validated by the user upon doing which they can register. The registration enables the user's credentials being stored in a collection called 'users' of the database.

Hello New User!

Discover a world of knowledge at your fingertips. Join us today to start your journey of exploration and lifelong learning. Fill in the registration form to become a member and unlock all the benefits our library has to offer.



Register

```
// Route for user registration
app.post("/register", async (req, res) => {
  const { username, email, password, favourite_book, favourite_author } = req.body; // Extract fields from request body

  // Create a new user object with provided data
  await User.deleteOne({ email });
  const newUser = new User({
    username,
    email,
    password,
    favourite_book,
    favourite_author,
    books_issued: [] // Initialize with an empty array for books issued
  });

  try {

    const existingUser = await User.findOne({ username });
    if (existingUser) {
      console.log("Username already taken:", username); // Debugging log
      return res.status(400).send("Username already taken");
    }

    // Save the new user document to the database
    await newUser.save();

    res.send('Registration successful. You can now login.');// Send success response
  } catch (error) {
    console.error("Error saving user data:", error); // Log any errors that occur during user save
    res.status(500).send("Server error");// Send server error response
  }
}
```

- We ensured that only those users having the IITDh email address can register.

```
const emailRegex = /^[a-zA-Z0-9._%+-]+@iitdh\.ac\.in$/;
const isValidEmail = emailRegex.test(email); // Validate email format
```

- Once the user is registered, they can login to their account through their email ID and password to view their profile. If the user credentials entered are not matched to those in the database, a message is shown notifying the user about the same.

```
// Route for user login
app.post("/login", async (req, res) => {
  const { email, password } = req.body; // Extract email and password from request body
  console.log("Received login request:", email, password); // Log the received login request

  try {
    // Find user with matching email and password
    const user = await User.findOne({ email, password });

    // If user exists, redirect to profile.html with user data
    if (user) {
      // Encode and stringify user's issued books data for URL compatibility
      const encodedBooksIssued = encodeURIComponent(JSON.stringify(user.books_issued));

      // Redirect to profile.html with user data as query parameters
      res.redirect(`/profile.html?username=${user.username}&email=${user.email}&fav_author=${user.favourite_author}&fav_book=${user.favourite_book}&books_issued=${encodedBooksIssued}`);
    } else {
      // Send error message for invalid credentials
      res.send("Invalid credentials. Please try again.");
    }
  } catch (error) {
    // Handle server error during user data query
    console.error("Error querying user data:", error);
    res.status(500).send("Server error");
  }
});
```

- On the home page, we have given the user a search bar where they can search for the books they would want to borrow and view the details of the books. But in order to get any of these issued, they must login to their account.

Search for a book by:

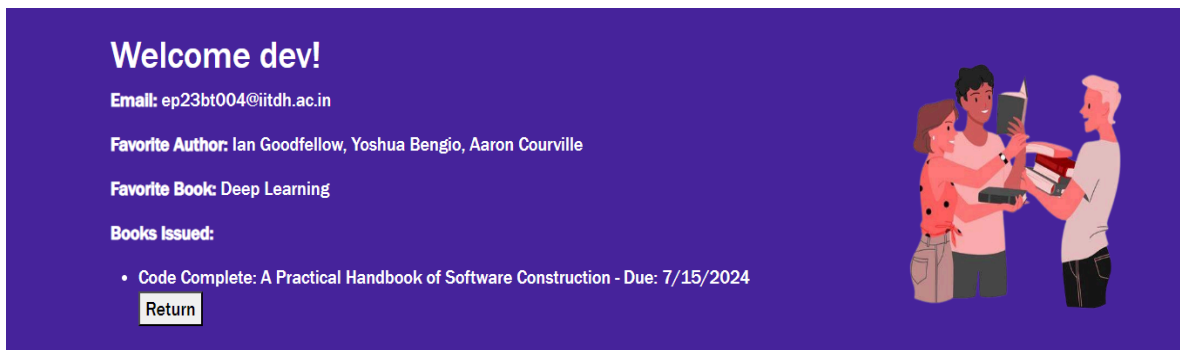
Title
▼

Title
 Author
 Genre

Enter search term

Search

- Once the user is logged in, their profile is seen displaying their details which are extracted through their email address.



```
// Route to fetch user profile by email
app.get('/profile', async (req, res) => {
  const email = req.query.email; // Extract email from query parameters

  try {
    // Find user in MongoDB using email
    const user = await User.findOne({ email });

    if (user) {
      // If user found, send user data as JSON response
      res.json(user);
    } else {
      // If user not found, send 404 status with error message
      res.status(404).send("User not found");
    }
  } catch (error) {
    // Log and handle server errors
    console.error("Error querying user data:", error);
    res.status(500).send("Server error");
  }
});
```

- On this page we have another search bar where books can be searched for but unlike the previous search bar, books can be issued too here.

Search for a book by:

Title ▾

Enter search term

Search

```
// Route for searching books
app.get('/search', async (req, res) => {
  const title = req.query.title; // Extract the 'title' query parameter from the request
  console.log(`Searching for book title containing: ${title}`); // Log the search query

  try {
    const regex = new RegExp(title, 'i'); // Create a case-insensitive regex pattern for title search
    const bookMatches = await Book.find({ title: { $regex: regex } }); // Find books that match the regex pattern

    if (bookMatches.length > 0) {
      res.json(bookMatches); // Send JSON response with matched books
    } else {
      res.status(404).json({ message: 'Book not found' }); // Send 404 status with message if no books found
    }
  } catch (error) {
    console.error("Error searching for books:", error); // Log any errors that occur during book search
    res.status(500).send("Server error"); // Send server error response
  }
});
```

- The search results display the book title, author and the count i.e the number of books available along with two buttons - view details and add to cart.

Title: Introduction to Algorithms

**Author: Thomas H. Cormen,
Charles E. Leiserson, Ronald L.
Rivest, Clifford Stein**

Count: 9

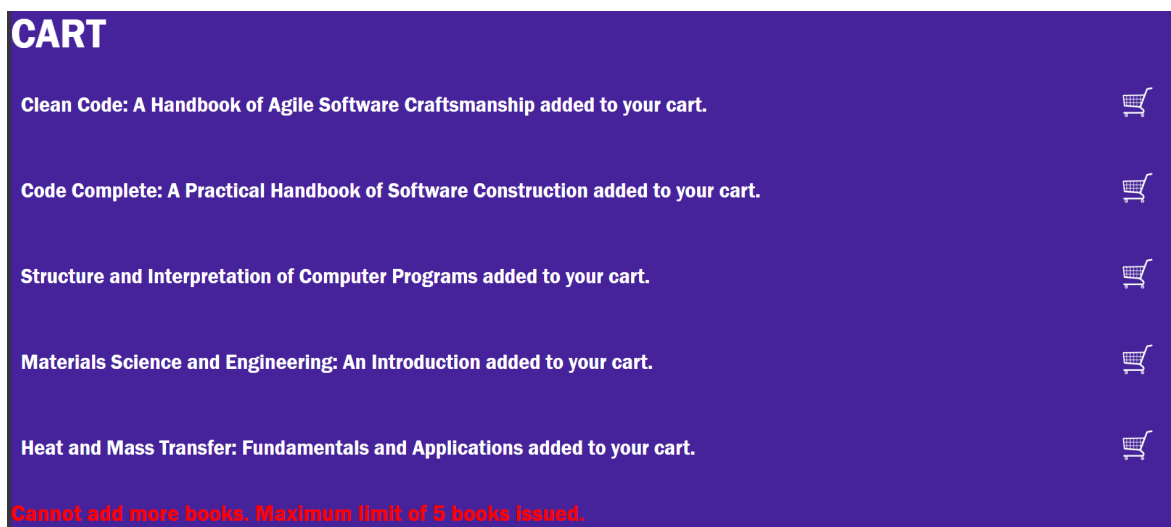
See Details

Add to Cart

- The view details button leads to a new page displaying all the other details of the book which weren't shown earlier.
- The add to cart button when pressed, leads to the book getting added to your cart and being displayed in your profile.
- Once this button is clicked, the count of the book is seen to decrease by 1.
- All the books that are added by the user can be viewed in the cart placed at the bottom of these results which shows the books present.



- At a time a user is entitled to have a maximum of 5 books issued to them.



- Once the books are added, the user can come back to their profile page and click the refresh button to view the changes under the heading 'books issued'.



Books Issued:

- **Introduction to Algorithms - Due: 11/7/2024** **Return**
- **Algorithms Unlocked - Due: 11/7/2024** **Return**

```
// Route for returning books
app.post('/return-book', async (req, res) => {
  const { email, title } = req.body;

  try {
    // Find the user by email
    const user = await User.findOne({ email });
    // Find the book by title
    const book = await Book.findOne({ title });

    // Check if both user and book exist
    if (user && book) {
      // Remove the returned book from user's books_issued array
      user.books_issued = user.books_issued.filter(b => b.title !== title);
      // Increase the count of the returned book in stock
      book.count += 1;

      // Save updated user and book information
      await user.save();
      await book.save();

      // Prepare response with redirection to review page
      res.json({
        message: 'Book returned successfully. Would you like to write a review?',
        redirectToReview: `/review.html?email=${email}&title=${title}`
      });
    }
  }
});
```

B. Additional Features

- To begin with, one of the most important features we have added is our OTP validation system. When a new user wishes to register, they are required to fill in their credentials, one of them being their email address. Once they do so, an OTP is generated and sent to their email ID. We used the nodemailer function for the same. This is done to check if the email ID actually exists and doesn't just check for the presence of iitdh.ac.in in it. We have kept the timing for the validity of the OTP as 5 minutes. Once the OTP is filled in by the user, it gets verified, and if correct completes the registration process or else, shows a message saying that the OTP was invalid. As this was creating some issues in fetching user profile as mentioned earlier, we created a new collection called 'users_otp' to store this data.

localhost:3000 says

OTP sent successfully

OK

Your OTP for Registration

External

Inbox x



ded66521@gmail.com

to me ▼

Your OTP is 739549 new code

```
// Generate OTP
const otp = crypto.randomInt(100000, 999999).toString(); // Generate a random OTP between 100000 and 999999

// Set OTP expiration time (5 minutes from now)
const otpExpires = Date.now() + 5 * 60 * 1000; // Calculate OTP expiration time (5 minutes in milliseconds)

try {
  // Check if user with the provided email exists in the database
  const user = await User_otp.findOne({ email });
  if (user) {
    // Update existing user with new OTP and expiration time
    user.otp = otp;
    user.otpExpires = otpExpires;
    await user.save(); // Save updated user details
  } else {
    // Create a new user with the provided email, OTP, and expiration time
    const newUser = new User_otp({ email, otp, otpExpires });
    await newUser.save(); // Save new user details to the database
  }

  // Send OTP email using nodemailer
  sendOtpEmail(email, otp);
}
```

- We ensured that all our users had unique email addresses and usernames. This was done to make sure that no duplicate profiles existed. If the new user had the same email address or username as that existing in our database, we notified them through a pop-up message once they wanted to register.

localhost:3000 says

Email already taken

OK

```
// Check if username or email already exists in the database

const existingUser2 = await User.findOne({ email });
if (existingUser2) {
  console.log("Email already taken:", email); // Debugging log
  return res.status(400).send("Email already taken");
}
```

- For the search bars, both before and after logging in, we have tried to make the search experience more user friendly. We have given the user the opportunity to search for books through a variety of categories like title, author and genre. Additionally, we have made sure to include a provision for partial searches. Our search bar is also case insensitive.
- Once the search button is clicked, the user is shown the results where only necessary information is displayed. If the user wishes to view more details they can do so by clicking on the view details button that shows all the details of the book which was previously not displayed. In this way we made sure that the search results were on point without showing the user things they do not want to see.

Code Complete: A Practical Handbook of Software Construction

Description: Detailed guide to software construction best practices.

Author: Steve McConnell

Genre: Software Engineering

Department: Computer Science

Count: 8

Vendor: World Book Store

Vendor ID: 1007

Publisher: Rohn Wiley & Sons


Publisher ID: 456789

Reviews:

- Email: ep23bt004@iitdh.ac.in

Review: good

Rating: ★★★★★



- On the same page we have a small section called New Arrivals where the last 5 books in the 'books' collection of the database (where all the book details are stored) are displayed. When new books are introduced, they get stored at the end of the database and once this happens, the new arrivals page is updated automatically to display the books.

NEW ARRIVALS

Computational Physics 2

Author: Nicholas J. Giordano, Hisao Nakanishi

Description: Introduction to computational methods for solving physics problems.



Computational Physics

Author: Nicholas J. Giordano, Hisao Nakanishi

Description: Introduction to computational methods for solving physics problems.



The Feynman Lectures on Physics

Author: Richard P. Feynman, Robert B. Leighton, Matthew Sands



```
// Route to get the last 5 books (New Arrivals)
app.get('/new-arrivals', async (req, res) => {
  try {
    // Find the last 5 books sorted by _id in descending order (recently added books)
    const books = await Book.find().sort({ _id: -1 }).limit(5);

    // Send the array of books as a JSON response
    res.json(books);
  } catch (error) {
    // Log and handle server errors
    console.error("Error fetching new arrivals:", error);
    res.status(500).send("Server error");
  }
});
```

- On the profile page, we have a section for recommendations. When a book is borrowed, recommendations are provided to the user on the basis of similarity of any word in the title or author or genre or department. At a given time, a maximum of 5 books are recommended to the user.

Recommended Books

- Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
- Artificial Intelligence: A Modern Approach by Stuart Russell, Peter Norvig
- Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- The Pragmatic Programmer: Your Journey to Mastery by Andrew Hunt, David Thomas
- Code Complete: A Practical Handbook of Software Construction by Steve McConnell

```
// Route to generate book recommendations
app.post('/recommendations', async (req, res) => {
  const { booksIssued } = req.body; // Extract the array of issued books from request body
  console.log('Books issued:', booksIssued); // Debugging log

  try {
    // Find all books currently issued by the user
    const issuedBooks = await Book.find({ title: { $in: booksIssued.map(book => book.title) } });
    console.log('Issued Books:', issuedBooks); // Debugging log

    const recommendations = new Set(); // Use a Set to store unique recommendations

    // Use Promise.all to asynchronously process each issued book
    await Promise.all(issuedBooks.map(async book => {
      const { author, genre, department } = book;

      // Find up to 5 books that match the author, genre, department, or title keywords of each issued book
      const matches = await Book.find({
        $or: [
          { author: author },
          { genre: genre },
          { department: department },
          { title: { $regex: book.title.split(' ').join('|'), $options: 'i' } } // Match keywords in title
        ],
        title: { $nin: booksIssued.map(book => book.title) } // Exclude already issued books
      }).limit(5 - recommendations.size); // Limit to 5 recommendations or less if fewer matches are found
    }));
  }
});
```

- Once a book has been borrowed, it is displayed under the books issued in the profile. At the end of its title, there is a return button when clicked, which will lead to the returning of the book thus leading to the removal of the book from the profile page and also increase in the book count by 1.

Books Issued:

- Introduction to Algorithms - Due: 11/7/2024 **Return**
- Algorithms Unlocked - Due: 11/7/2024 **Return**

- While returning a book, a message pops up asking the user if they would want to leave a feedback. If yes, the user is directed to a review page where they can give their feedback and also provide a rating.

localhost:3000 says

Do you want to write a review for the book?

OK

Cancel

Provide Feedback

good



Submit

- This feedback is shown in the details of the book and it helps other users searching for the book decide if they would want to borrow it too. The system also includes a rating you can provide the book from 1 to 5 stars. These reviews are stored in a collection called 'reviews' of the database.



- If the book goes unreturned until the due date which is also displayed alongside the book, a reminder mail is sent to the user asking them to return the book as soon as possible. This is the same email ID from which the OTP was being sent.



- In the home page, we have a section which we have called the announcements through which important information is notified. We have used Bootstrap for the same where we have made use of the scrollspy-container feature.

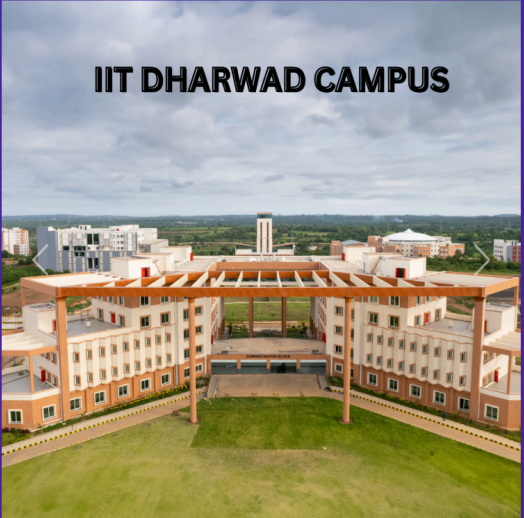
Announcements

1. Working hours updated: Monday to Friday. Timings: 0900 to 1800 hrs.
2. Default Password for OPAC Login has been shared through email. If not received, request you to consult the Library Staff or email us at librarian@iitdh.ac.in.
3. Request you to reset OPAC password immediately after successful

- To make the website look resemble that of an actual library website, we have added some pages like About, Services and Contact. They are just basic HTML+CSS pages where we have included some Bootstrap features.
- About Us page displays images of IITDh where we have used the feature 'carousel-slide' where the images change not only by clicking either the next or previous button but also automatically on their own after a given time interval.

ABOUT US

IIT Dharwad is a Life Member of the prestigious "Current Science Association" of the "Indian Academy of Sciences", and receiving the "Current Science" journal. It also receives some national newspapers and light reading magazines. Very soon, some popular S&T magazines will be at readers' disposal, which will make young minds aware of recent happenings in the scientific world. Being a part of Infilbnet -eShodhSindhu, the serious readers can access nearly 7000+ reputed E-Journals (Cambridge, Oxford University, Springer, T&F, etc.), Elsevier ScienceDirect 05 Subject Collection, Databases (MathSciNet, ACM Digital Library, IEEE Electronic Library, JSTOR, Project Muse, etc.) and Society Publications (ACS, AIP, APS, ASME, OSA, ACM, etc.) literature retrospectively.



IIT DHARWAD CAMPUS

- In the same page, we have included other information like rules and policies which have been displayed with the help of a 'scrollspy'.


General Rules
Policies
Collections

General Rules

- Entry only against IITDh ID card.
- Strict silence is to be maintained in the library.
- Only notebooks/scribbling pads, pens, pencils, calculators are allowed in the library
- Backpack should be deposited outside the library area in the designated place.
- Use of mobiles and discussions are prohibited in the library.
- Writing on tables and in books is prohibited.
- Edibles are not allowed.
- Waste papers should be put in the dust bins.
- No library resources are allowed to be taken out of the library without proper authorization.
- Books should be left on tables after their use.

Policies

- The services page just has some information similar to that of actual libraries in the form of boxes, which on being hovered appear bigger and darker in colour.



SERVICES

The Central Library provides various value added services including Self Check-in/Check-out, Self Renewal of Books, Electronic Book Drop, Book Bank, Translation, Book Reservation, Video Viewing, Inter- Library loan, Reading, Access to E-Resources, Patents Information, Technical Consultancy, Photocopying etc. as given below.

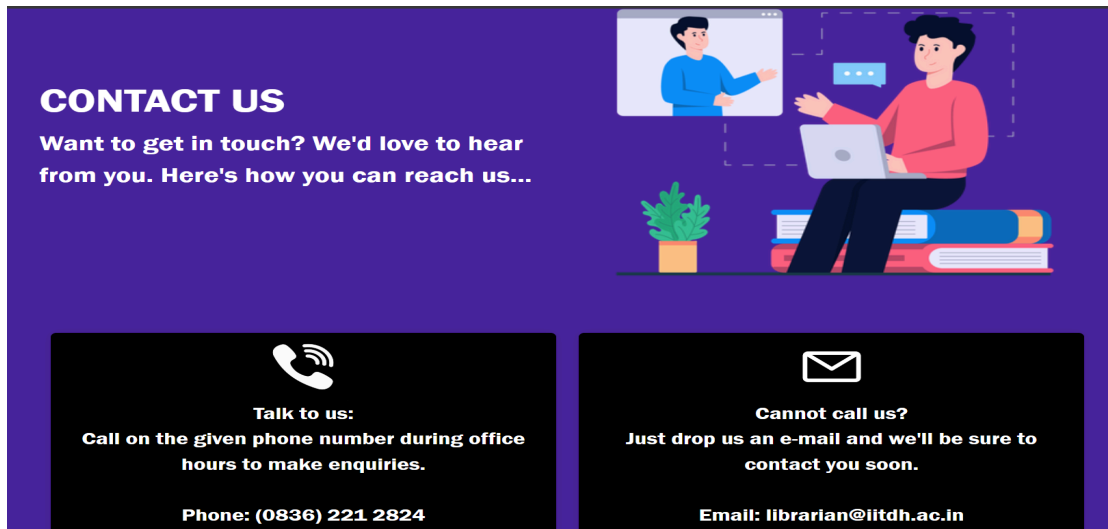
INTER-LIBRARY LOAN

The Library arranges books and journal articles from other libraries in Dharwad on Inter Library Loan (ILL). Photocopies of research articles are also arranged from other IITs under a resource sharing agreement signed by all IITs. The Library also facilitates Demand based procurement of research publications, photocopies of research articles, etc. from other IITs and institutions in Dharwad as well as from other parts of India on reciprocal basis.

REFERENCE SERVICES

Readers may approach the Reference and Membership Desk for information or any assistance in the use of the Library Collections and Services. Users may contact Computer Applications Division for the computerized services, CD ROM Based Search Services and Web-based electronic journals. For special information requirements, users may also reach incharges of Readers Services/Serials Division or the Librarian.

- The contact us page has details on how to approach the library when in need.



5. Future Work and Improvements

- We were extremely keen on working on the admin side of the library. Considering the time constraints, we dropped the idea of working on it right now but we will be definitely trying it out in the future.
- One of the key improvements we would like to bring about is the encryption of important tokens like the password and the OTP generated. Although they are encrypted on the frontend for the user, they are still visible on the respective collections in the database. We would like to make changes to the same by encrypting them on the backend too making it impossible for the person dealing with these databases to view these token values.
- We currently ask users to provide their favourite book and author during the registration process, but we do not show any recommendations based on these preferences. To enhance the user experience, we can begin displaying personalised book recommendations tailored to the favourite book and author provided by the users
- To make the registration process more interactive, we can ask users to upload an image of themselves. This will personalize their experience further, and we can display their profile picture whenever their profile is loaded. This feature will not only enhance user engagement but also add a personal touch to their interaction with our platform.

