Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Computer Science and Engineering

Department of Computer Engineering and Technology

Third Year B. Tech. CSE (Cybersecurity and Forensics)

CSF3PM01A: Full Stack Development Laboratory

Resource Knowledge Hub - Report

Roll No.	Name	PRN	Student Signature
03	Yashica Sanap	1032222033	
06	Sindhura Gulhane	1032230397	
18	Strelisha Lewis	1032231040	
28	Rithin Shaju	1032231269	

Course Name: **Full Stack Development**
Course Code: **CSF3PM01A**
Submission Date: **09.11.2025**
Submitted to: **Prof. Sagar Apune**
Signature:

1. ABSTRACT

“**Resource Knowledge Hub”** is a full-stack web application designed to centralize and simplify access to academic and research materials for both learners and educators. The platform allows users to register, log in, upload, browse, and download learning resources such as PDFs, Word documents, and text files.

The aim of the project is to reduce resource fragmentation among students and educators by providing an easy-to-use, secure, and scalable web portal. The system uses a **React-based frontend** to ensure a responsive and dynamic user interface and a **Node.js + Express backend** for seamless API handling and data management. Files are stored locally, while metadata is maintained in a **JSON database**.

The project successfully demonstrates a minimal yet complete full-stack development workflow incorporating authentication, file handling, and dynamic content rendering.

2. INTRODUCTION

2.1 Background and Motivation

In most academic environments, study materials are scattered across various drives, folders, and messaging platforms. The fragmentation makes it difficult for students and faculty to locate important resources efficiently and quickly.
This project addresses the issue by offering a **centralized and unified platform** for uploading, browsing, and downloading academic content with ease while maintaining efficiency.

2.2 Problem Statement

Students and faculty currently lack an accessible and organized system for sharing study materials, resulting in redundancy and inefficiency.
The proposed system solves this problem by providing structured uploading, categorization, and retrieval of files.

2.3 Project Objectives

- To design and implement a full stack web-based system for managing academic resources.
- To integrate **user authentication** for secure uploads and downloads.

- To develop a **responsive and user-friendly web application** using *React*, supported by *Node.js and Express* for backend API management, with reliable **data persistence** through *JSON-based storage*.

2.4 Scope and Limitations

Scope:

- User registration and authentication
- File uploads and downloads
- Search and filtering functionality
- Resource management interface

Limitations:

- No real-time collaboration yet
- Uses local file and JSON storage instead of cloud-based solutions (can be upgraded in future versions)

3. LITERATURE REVIEW

3.1 Overview

The growing digitization of education has made academic resource sharing more important than ever. However, most available platforms are either too general-purpose or too complex for small academic environments. The Resource Knowledge Hub aims to bridge this gap by offering a **lightweight, secure, and easy-to-use** system for uploading, browsing, and downloading learning materials using modern web technologies.

3.2 Review of Related Work

1. **Moodle – Open-Source Learning Platform (Dougiamas & Taylor, 2003)**

Moodle, an open-source LMS that supports course creation, assignments, and resource sharing was introduced by Dougiamas and Taylor. While providing a strong educational management features, Moodle is **heavyweight and complex to deploy**, requiring full server and database setup.
In comparison, the Resource Knowledge Hub offers a **simpler, file-centered platform** focused solely on resource organization and sharing without the overhead of an LMS.

2. **Google Drive and Dropbox – Cloud Storage Platforms (Google, 2023; Microsoft, 2023)**

Services like **Google Drive** and **Dropbox** provide powerful file storage and collaboration tools but are **not tailored for academic use**. It lacks subject-based organization, metadata tagging, and structured search relevant to educational resources.
Contrasting to which the "Resource knowledge Hub" includes **categorization by topic, author, and type**, making it faster and more meaningful for retrieval of study materials.

3. **Lightweight CMS using MERN Stack (Kumar & Singh, 2021)**

Kumar and Singh demonstrated that lightweight CMS platforms using the **MERN stack** *(MongoDB, Express, React, Node.js)* can improve modularity and performance. The research supports the **architecture adopted in our project**, though our system replaces MongoDB with **JSON storage** for simplicity and focuses on **academic resources** instead of general content management.

4. **Cloud-Based Educational Resource Management (Zhou et al., 2022)**

Zhou et al. proposed a **cloud-based resource system** which uses metadata extraction and AI for intelligent content classification. Although advanced, it requires complex infrastructure and higher maintenance costs.
Our project, alternatively, maintains a **minimal and deployable design** suitable for local or institutional use while leaving room for future cloud and AI integration.

3.3 Summary

Overall, the existing systems either emphasize **broad functionality** (e.g., Google Drive, Moodle) or **high technical complexity** (e.g., cloud-based ML models). The **Resource Knowledge Hub** combines the best of both worlds — **simplicity, usability, and academic relevance** — making it ideal for small institutions and collaborative student groups.

4. METHODOLOGY

4.1 System Overview

The system follows a **client-server architecture** with a React-based frontend and a Node.js + Express backend, communicating through REST APIs.

4.2 Tech Stack

Layer	**Technology**	**Purpose**
Frontend	React.js, Bootstrap	UI components, routing, responsive design
Backend	Node.js, Express.js	RESTful API creation, authentication, business logic
Database	JSON file (db.json)	Data persistence for users and resources
File Handling	Multer	File upload and storage management
Authentication	JWT (JSON Web Token)	Secure user login and access control

4.3 System Architecture

Frontend (React)

- **Components:** Home, Explore, Upload, Login, Register, Navbar
- **Routing:** Implemented using *react-router-dom*
- **API Calls:** Managed via *Axios*
- **UI Framework:** Bootstrap for layout and responsiveness

Backend (Node.js + Express)

Routes Implemented:

- /api/auth/register – User registration
- /api/auth/login – User authentication
- /api/resources – Upload and retrieve resources
- /api/resources/:id/download – Download files

Middleware:

- JWT-based authentication for protected routes

Storage:

- Files stored in /uploads folder
- Metadata managed in db.json

4.4 Development Approach

The development follows an **Agile methodology**, emphasizing iterative progress and testing.

Phases:

1. UI Prototyping
2. API Creation and Testing
3. Frontend–Backend Integration
4. UI Styling and Form Validation
5. Final Testing and Presentation

5. RESULTS AND DISCUSSION

5.1 Results

The system successfully implemented the following features:

- Secure **user registration and login** using JWT
- File **upload and download** functionalities
- **Search and filter** features (by title, author, topic)
- **Responsive UI** using React and Bootstrap

5.2 Discussion

Achievements:

- Simple and intuitive user experience
- Robust upload process supporting multiple file formats
- Efficient local storage for testing and small-scale deployment

Challenges Encountered:

- Path inconsistencies when files were moved between systems
- Port configuration issues between Vite (React) and Express
- CORS (Cross-Origin Resource Sharing) errors during API integration

Solutions:

- Introduced relative path handling
- Adjusted development proxy settings
- Configured CORS middleware on the server

6. CONCLUSION

The **Resource Knowledge Hub** achieved its main objective of providing a **secure and efficient platform** for sharing and accessing academic resources.
The use of a **MERN-like stack (React + Express + JSON)** effectively demonstrated key full-stack concepts, including authentication, file handling, and frontend-backend integration.

Key Achievements

- Integrated RESTful APIs with a dynamic React frontend
- Implemented secure upload/download mechanisms
- Delivered a clean, pastel-themed, mobile-friendly UI

Future Enhancements

- Migrate from JSON storage to **MongoDB Atlas** for scalability
- Add **role-based access control** (Admin, Faculty, Student)
- Introduce **file preview and rating** features
- Implement **email notifications** for collaboration and updates

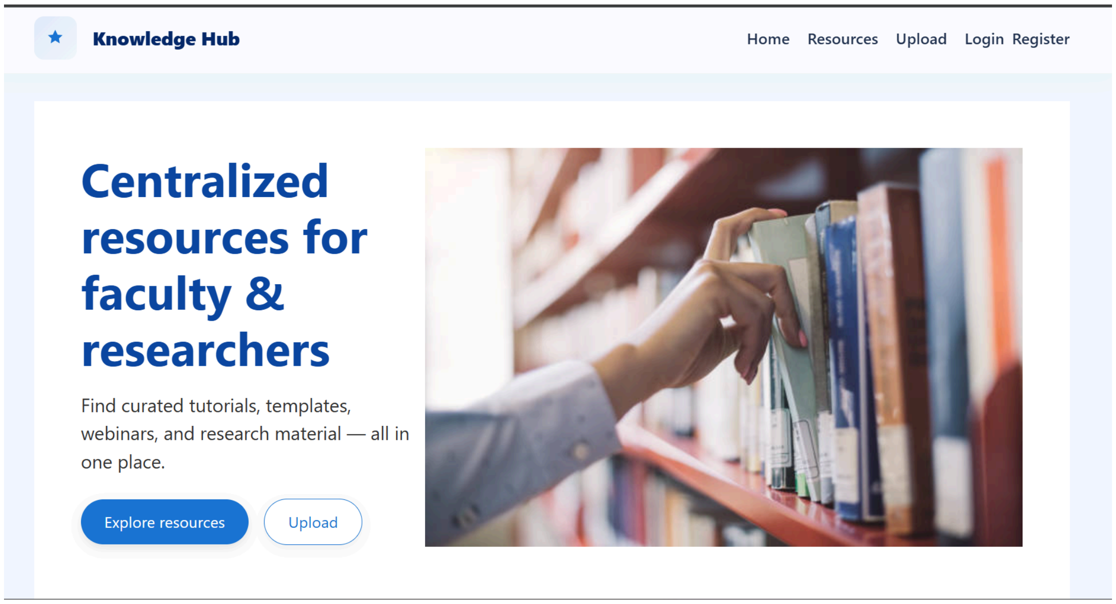
7. REFERENCES

1. Dougiamas, M., & Taylor, P. C. (2003). *Moodle: Using learning communities to create an open source course management system*. Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications, 171–178.
2. Google. (2023). *Google Drive*. Retrieved from https://www.google.com/drive/
3. Microsoft. (2023). *Dropbox cloud storage and file sharing*. Retrieved from https://www.dropbox.com/
4. Kumar, A., & Singh, R. (2021). *Design and implementation of a lightweight content management system using the MERN stack. International Journal of Computer Applications*, 183(30), 22–27. https://doi.org/10.5120/ijca2021921356
5. Zhou, L., Wang, X., & Li, H. (2022). *Cloud-based educational resource management system based on metadata and artificial intelligence. Journal of Cloud Computing and Education Technology*, 12(4), 55–68. https://doi.org/10.1016/j.cloudedu.2022.04.005

8. APPENDICES

A. Images

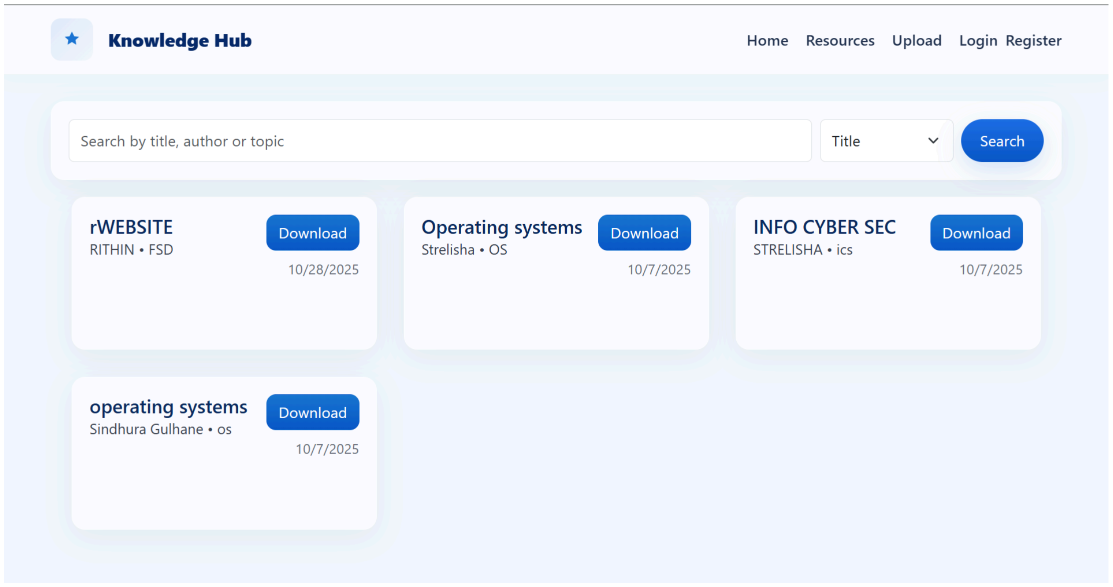
- 8.A.1- Landing Page



Centralized resources for faculty & researchers

Find curated tutorials, templates, webinars, and research material — all in one place.

- 8.A.2- Resource Library



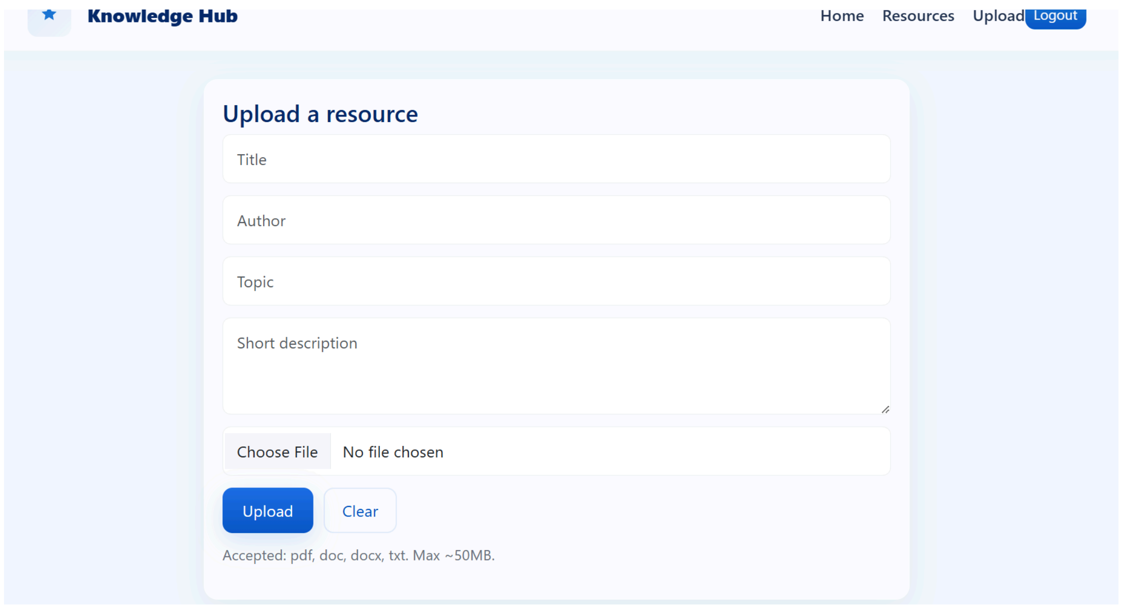
rWEBSITE
RITHIN • FSD
10/28/2025

Operating systems
Strelisha • OS
10/7/2025

INFO CYBER SEC
STRELISHA • ics
10/7/2025

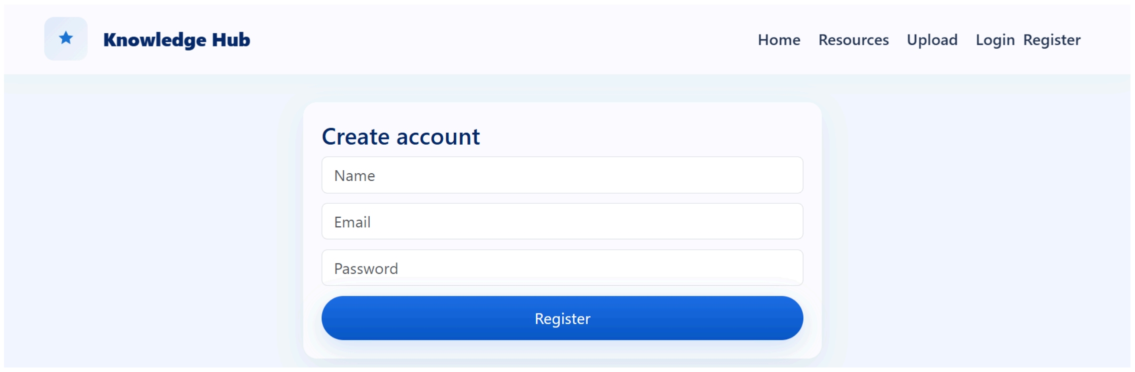
operating systems
Sindhura Gulhane • os
10/7/2025

- 8.A.3- Upload Page(Allows uploads only after Login)


Knowledge Hub
Home Resources Upload Logout

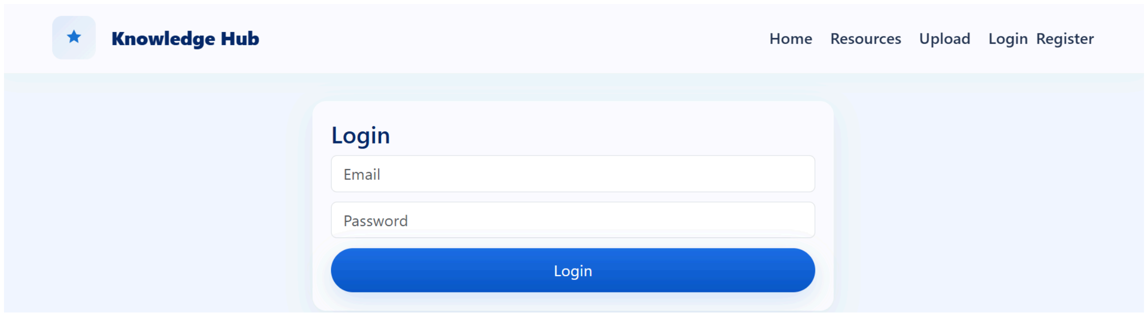
Upload a resource
Title
Author
Topic
Short description
Choose File No file chosen
Upload Clear
Accepted: pdf, doc, docx, txt. Max ~50MB.

- 8.A.4-Register Page


Knowledge Hub
Home Resources Upload Login Register

Create account
Name
Email
Password
Register

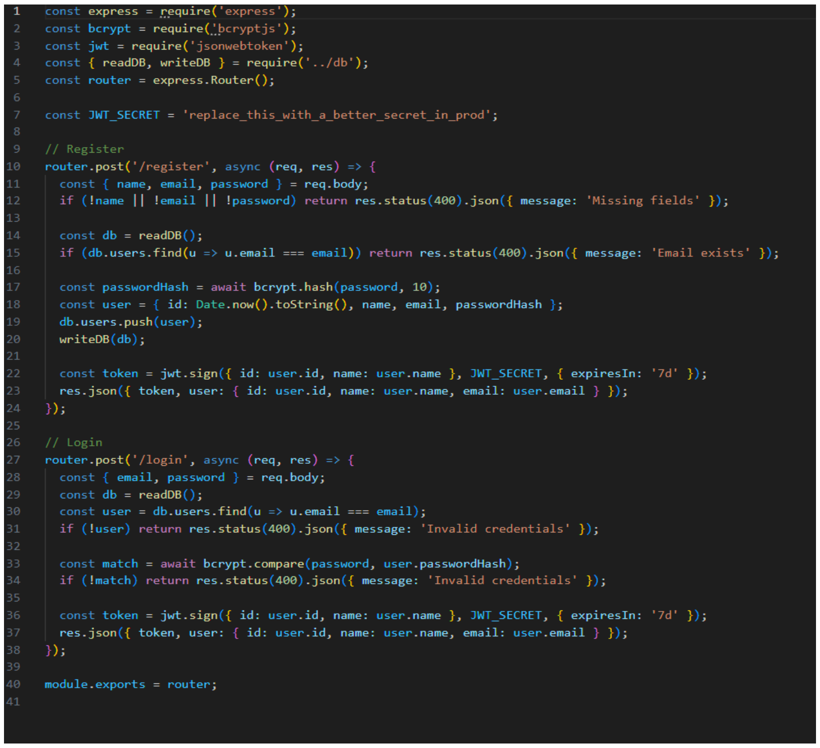
- 8.A.5-Login Page


Knowledge Hub
Home Resources Upload Login Register

Login
Email
Password
Login

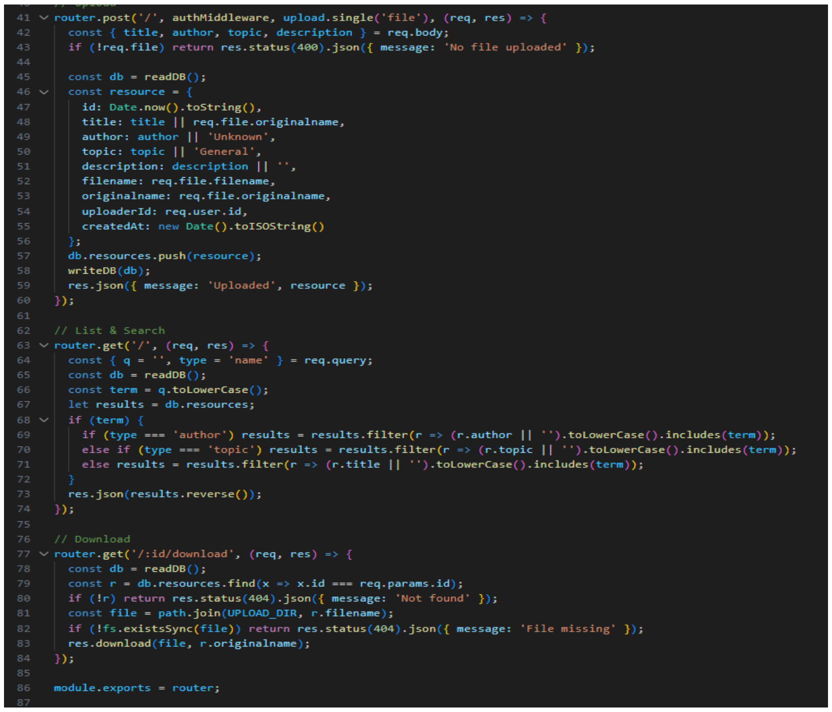
B. Code Snippets

- 8.B.1- Authentication Code



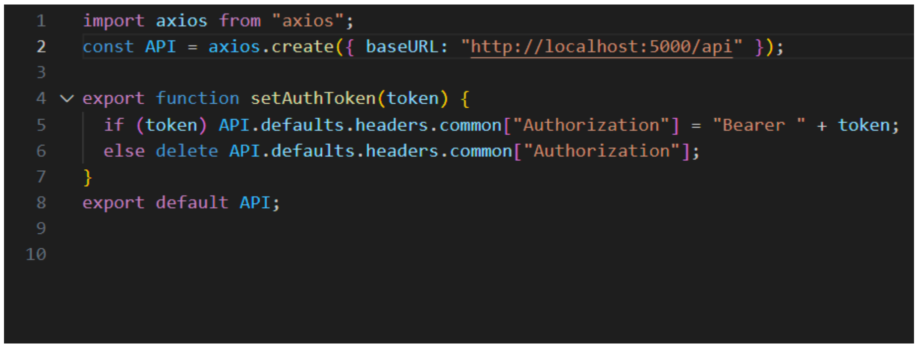
```javascript
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { readDB, writeDB } = require('../db');
const router = express.Router();

const JWT_SECRET = 'replace_this_with_a_better_secret_in_prod';

// Register
router.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  if (!name || !email || !password) return res.status(400).json({ message: 'Missing fields' });

  const db = readDB();
  if (db.users.find(u => u.email === email)) return res.status(400).json({ message: 'Email exists' });

  const passwordHash = await bcrypt.hash(password, 10);
  const user = { id: Date.now().toString(), name, email, passwordHash };
  db.users.push(user);
  writeDB(db);

  const token = jwt.sign({ id: user.id, name: user.name }, JWT_SECRET, { expiresIn: '7d' });
  res.json({ token, user: { id: user.id, name: user.name, email: user.email } });
});

// Login
router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const db = readDB();
  const user = db.users.find(u => u.email === email);
  if (!user) return res.status(400).json({ message: 'Invalid credentials' });

  const match = await bcrypt.compare(password, user.passwordHash);
  if (!match) return res.status(400).json({ message: 'Invalid credentials' });

  const token = jwt.sign({ id: user.id, name: user.name }, JWT_SECRET, { expiresIn: '7d' });
  res.json({ token, user: { id: user.id, name: user.name, email: user.email } });
});

module.exports = router;
```

- 8.B.2-Upload Code



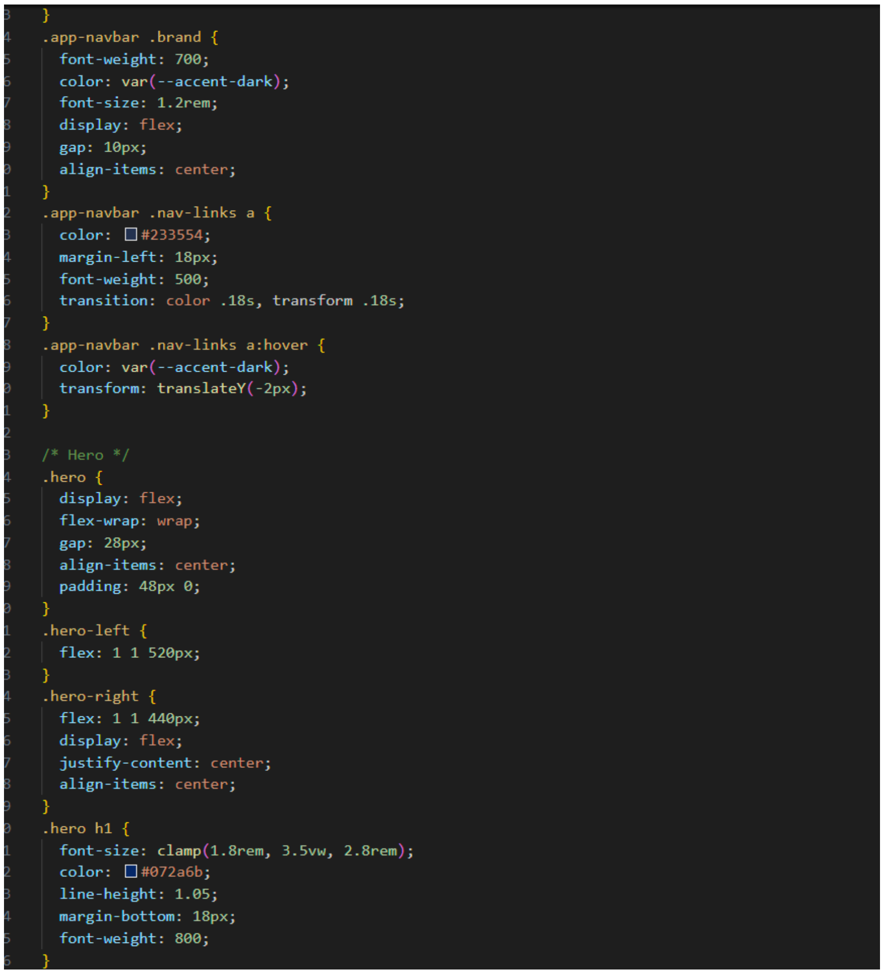
```javascript
router.post('/', authMiddleware, upload.single('file'), (req, res) => {
  const { title, author, topic, description } = req.body;
  if (!req.file) return res.status(400).json({ message: 'No file uploaded' });

  const db = readDB();
  const resource = {
    id: Date.now().toString(),
    title: title || req.file.originalname,
    author: author || 'Unknown',
    topic: topic || 'General',
    description: description || '',
    filename: req.file.filename,
    originalname: req.file.originalname,
    uploaderId: req.user.id,
    createdAt: new Date().toISOString()
  };
  db.resources.push(resource);
  writeDB(db);
  res.json({ message: 'Uploaded', resource });
});

// List & Search
router.get('/', (req, res) => {
  const { q = '', type = 'name' } = req.query;
  const db = readDB();
  const term = q.toLowerCase();
  let results = db.resources;
  if (term) {
    if (type === 'author') results = results.filter(r => (r.author || '').toLowerCase().includes(term));
    else if (type === 'topic') results = results.filter(r => (r.topic || '').toLowerCase().includes(term));
    else results = results.filter(r => (r.title || '').toLowerCase().includes(term));
  }
  res.json(results.reverse());
});

// Download
router.get('/:id/download', (req, res) => {
  const db = readDB();
  const r = db.resources.find(x => x.id === req.params.id);
  if (!r) return res.status(404).json({ message: 'Not found' });
  const file = path.join(UPLOAD_DIR, r.filename);
  if (!fs.existsSync(file)) return res.status(404).json({ message: 'File missing' });
  res.download(file, r.originalname);
});

module.exports = router;
```

- 8.B.3-Axios Config



```js
import axios from "axios";
const API = axios.create({ baseURL: "http://localhost:5000/api" });

export function setAuthToken(token) {
  if (token) API.defaults.headers.common["Authorization"] = "Bearer " + token;
  else delete API.defaults.headers.common["Authorization"];
}
export default API;
```

- 8.B.4-CSS



```css
}
.app-navbar .brand {
  font-weight: 700;
  color: var(--accent-dark);
  font-size: 1.2rem;
  display: flex;
  gap: 10px;
  align-items: center;
}
.app-navbar .nav-links a {
  color: #233554;
  margin-left: 18px;
  font-weight: 500;
  transition: color .18s, transform .18s;
}
.app-navbar .nav-links a:hover {
  color: var(--accent-dark);
  transform: translateY(-2px);
}

/* Hero */
.hero {
  display: flex;
  flex-wrap: wrap;
  gap: 28px;
  align-items: center;
  padding: 48px 0;
}
.hero-left {
  flex: 1 1 520px;
}
.hero-right {
  flex: 1 1 440px;
  display: flex;
  justify-content: center;
  align-items: center;
}
.hero h1 {
  font-size: clamp(1.8rem, 3.5vw, 2.8rem);
  color: #072a6b;
  line-height: 1.05;
  margin-bottom: 18px;
  font-weight: 800;
}
```