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 (DCET)

Full Stack Development Project
CSF Panel-A

PRNNO:1032230397 Name:SINDHURA GULHANE
PRNNO:1032231040 Name:STRELISHA LEWIS
PRNNO:1032231269 Name:RITHIN SHAJU
PRNNO:1032222033 Name:YASHICA SANAP

Problem Statement & Proposed Solution

PROBLEM STATEMENT-

In academic environments, learning resources are often scattered across multiple platforms such as personal drives, WhatsApp groups, and classroom folders. This leads to duplication, difficulty in searching, and lack of proper organization. Students and faculty waste valuable time locating study materials instead of focusing on learning. Furthermore, most existing storage solutions are general-purpose and do not cater specifically to educational needs like topic-wise categorization or author-based filtering. Hence, there is a need for a centralized, secure, and easy-to-use platform where students and educators can upload, manage, and access academic resources efficiently.

PROPOSED SOLUTION-

- Develop a web-based centralized repository where users can upload, browse, and download educational resources.
- Implement secure authentication (JWT) to restrict uploads to registered users.
- Provide search and filtering by title, author, and topic.
- Use a responsive React interface for seamless user experience.
- Use Node.js + Express backend for RESTful APIs and file management (via Multer).
- Store uploaded files locally and maintain metadata in a lightweight JSON database.

Process Flow Diagram & System Architecture

1)User Registers / Logs In

JWT Token generated

2)Frontend (React)

Sends HTTP request via Axios

3)Backend (Express)

validates JWT→ handles logic

4)Multer Middleware

stores uploaded file in `/uploads/`

5)db.json

stores file metadata (title, author, topic, filename)

6)Frontend (Explore Page)

fetches resources list → displays as cards

7)User Clicks Download

backend sends file using `res.download()`

Technical Stack with Reasoning

Layer	Technology	Reason for Selection
Frontend	React.js	Modular, responsive UI; efficient rendering
Styling	Bootstrap + Custom CSS	Quick, clean design with pastel theme
Backend	Node.js + Express.js	Simplified API creation; handles multiple routes easily
Database	Local JSON File	Lightweight, easy for mini projects; no setup overhead
Middleware	Multer	Efficient file upload handling
Security	JWT + bcrypt.js	Token-based secure authentication & password hashing
Networking	CORS	Enables frontend-backend communication on different ports

Challenges Faced during development if Any

- CORS errors while connecting React (5173) to backend (5000).
  → Fixed by enabling app.use(cors()).

- File path issues after changing project folder locations.
  → Resolved by using path.join(__dirname, "..", "uploads").

- Multer version mismatch (1.4.5).
  → Fixed by installing multer@1.4.5-lts.1.

- UI responsiveness and layout breaking after styling updates.
  → Adjusted flex/grid CSS and Bootstrap utility classes.

- Token loss on refresh.
  → Fixed by persisting JWT token in localStorage and re-setting headers on load.

Limitation and scope of the project with future projection extension

Limitations

- Currently uses a local JSON file instead of a full database.
- No admin or role-based permissions (everyone equal access).
- Lacks real-time collaboration or comment system.
- File previews not supported (only download).

Scope / Future Enhancements

- Integrate MongoDB Atlas for scalable cloud storage.
- Deploy backend on Render / Vercel for public access.
- Add role management (Admin, Faculty, Student).
- Implement file previews and thumbnails.
- Enable cloud backup using AWS S3 or Firebase Storage.
- Add user analytics and favorites feature.

Working Demo