**TECHNICALACUMEN**

**Project Overview**

**TechnicalAcumen** is a modern, interactive web platform designed to empower learners, tech enthusiasts, and professionals with up-to-date resources on Artificial Intelligence, Machine Learning, and the latest industry trends. The site brings together curated learning paths, real-time tech news, hands-on coding blogs, and a vibrant community chatroom all under a unified, user-friendly interface.

Our mission: make an advanced learning site to learning emerging concepts and industry knowledge accessible, organized, and engaging for everyone from beginners to experts.

**Key Features**

* **Curated AI/ML Learning Modules:** Organized by topic, with resources, guides, and interactive tutorials.
* **Auto-Updating Industry News:** Real-time tech and AI news sections powered by RSS feeds.
* **Explore Page:** One-stop hub to browse courses, blogs, podcasts, and trending content.
* **Tech Blogs & Podcasts:** Categorized blogs and podcast carousels for in-depth learning and inspiration.
* **Community Chatroom:** Live chat and discussion forum, inspired by Discord, for users to interact, share, and learn together.
* **Calendar Feature:** Integrated calendar for scheduling, upcoming events, and training sessions.
* **Leaderboard & Feedback:** Gamified features to encourage active participation and gather user insights.
* **AI Chatbot (Lex):** AWS Lex-powered assistant for instant help, Q&A, and platform guidance.
* **AWS Chime Integration:** Live classroom sessions, webinars, and real-time communication using Amazon Chime.
* **SharePoint Classroom Materials:** Seamless access to classroom training materials via SharePoint integration.
* **Modern UI/UX:** Neon-inspired, dark-themed design for a seamless and visually striking experience.

**Backend & Automation**

* **REVA Automation:** Used Chime and LEX services for interactive chatroom and chatbot.
* **Database:** **PostgreSQL** for robust, scalable data storage and user management.
* **API/Server:** **Node.js** backend powering core APIs and integrations.
* **Security & Access Control:** Role-based permissions and secure access for all users.
* **CI/CD:** Automated deployments for fast, reliable updates.

**Tech Stack**

**Frontend:**

* **React.js** – Core framework for building dynamic user interfaces
* **HTML5 & CSS3** – Custom styles with modern responsive layouts
* **JavaScript (ES6+)** – Logic, interactivity, and integrations
* **Tailwind CSS** – Utility-first CSS for fast, consistent UI design

**Backend:**

* **Node.js** – Server-side application logic and API endpoints
* **Express.js** – Routing and middleware for RESTful services
* **PostgreSQL** – Relational database for user data, posts, and platform content

**Cloud & Automation:**

* **Amazon Lex** – Conversational AI for the integrated chatbot
* **Amazon Chime SDK** – Real-time communication (live chat, video calls)

**Integrations:**

* **SharePoint** – Access and embed classroom training materials directly on the site
* **RSS Feeds** – Auto-updating industry news, blogs, and podcasts

**Features & Libraries:**

* **FullCalendar** – Interactive calendar and event scheduling
* **Socket.io** – Real-time chat updates in community forum
* **JWT/OAuth** – Secure authentication and role management

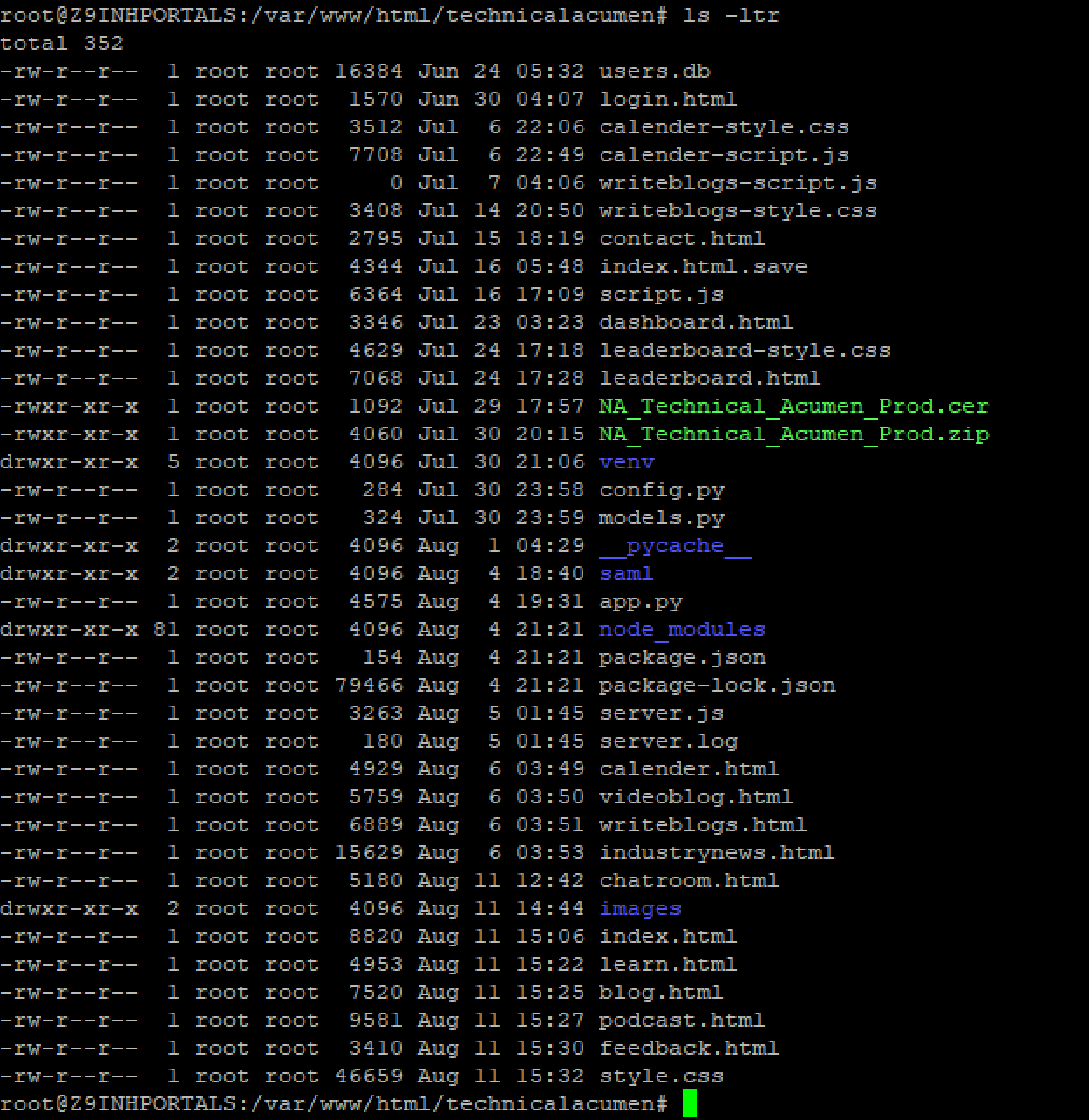
**DevOps & Deployment:**

* **Git & GitHub** – Version control and team collaboration

**Folder Structure**

Below is a typical folder structure for the **TechnicalAcumen** project inside the server.

Under “cd /var/www/html/technicalacumen” all the required files for the technicalacumen is available



**Key Code Snippets:**

**TechnicalAcumen – Detailed Project File Structure**

Root Directory: /var/www/html/technicalacumen

This folder now contains the entire website — frontend pages, backend scripts, stylesheets, images, and configurations.

**1️.** **Frontend – HTML Pages (User Interface)**

These are the visible pages users see in their browser. If you need to change page content, you edit these files:

* index.html – The homepage/landing page of TechnicalAcumen. Contains the hero section, navigation menu, and intro text.
* login.html – The user login form. Connects to backend authentication scripts.
* learn.html – Page listing AI/ML courses and categorized learning sections.
* writeblogs.html – Blog editor page. Lets users write and publish text-based blogs.
* videoblog.html – Displays technical videos/blogs fetched from YouTube or other sources.
* podcast.html – Podcast listing page with embedded Spotify players.
* feedback.html – Page for users to submit feedback via a form.
* industrynews.html – Tech news page with RSS feed integration.
* chatroom.html – Real-time chatroom/forum-style discussion page.
* leaderboard.html – Shows user rankings (possibly from gamified points).
* contact.html – Contains contact form and team contact info.
* blog.html – Lists published blog articles.

**2️. CSS Files (Design & Styling)**

These files control how the website looks (colors, layouts, fonts).

* style.css – Main stylesheet applied across the entire site.
* calender-style.css – Styles for any calendar/date picker components.
* writeblogs-style.css – Styles specifically for the Write Blogs page.
* leaderboard-style.css – Styles specifically for the Leaderboard page.

**3️. JavaScript Files (Frontend Logic)**

These files handle interactivity and dynamic page updates in the browser.

* script.js – Main JavaScript for general page functions (menus, animations, API calls).
* calender-script.js – Code to control calendar/date selector.
* writeblogs-script.js – Handles blog creation form submission, image uploads, and saving to database.

**4️.Backend Files (Server-Side Logic)**

These files run on the server to handle data storage, processing, and API requests.

* app.py – Python backend application (likely Flask). May handle blog storage, chatroom messages, or API endpoints.
* config.py – Stores backend configurations like DB paths, API keys, and settings.
* models.py – Python file defining database tables and structures.
* server.js – Node.js backend server script, possibly handling chatroom or API requests.
* server1.js – An alternate or backup Node.js server configuration.

**5️.Data & Database**

* users.db – SQLite database storing user details, blog posts, leaderboard points, etc.

**6️.Config & Dependency Files**

* package.json – Node.js project metadata & dependency list.
* package-lock.json – Locked dependency versions for Node.js.
* venv/ – Python virtual environment containing Python packages.
* node\_modules/ – Installed Node.js packages.

**7️.Static Assets**

* images/ – All images used on the website. Now hosted locally instead of GitHub.
* NA\_Technical\_Acumen\_Prod.cer – SSL certificate file for HTTPS.
* NA\_Technical\_Acumen\_Prod.zip – Backup archive containing SSL certificate and possibly private key.

**Future Works for TechnicalAcumen**

**1️. Chatroom + Chatbot Integration**

Current: Layout is ready, setup is done, but there are errors stopping it from fully working.

Work to be done:

Fix the errors so messages send and receive without issues.

Make sure both chatroom and chatbot can run together without interfering.

Test sending and receiving messages from both sides.

**2️. SSO (Single Sign-On) Login**

Current: Configuration files are already in the server, but the site is not connecting to SSO.

Work to be done:

Check all links, login addresses, and certificates are matching.

Fix any mismatch so SSO login works and takes the user into the site.

**3️. Search Bar**

Current: Search bar design is ready but doesn’t work yet.

Work to be done:

Connect the search bar so it can find pages or content on the site.

Show suggestions when the user types.

**4️. Feedback Form (Backend)**

Current: Feedback page is there, but messages are not stored anywhere.

Work to be done:

Save feedback submissions into a file or database.

Show a success message after feedback is sent.

**5️. Teams Calendar Integration**

Current: Calendar page is ready, but not linked to Teams.

Work to be done:

Connect the calendar to Microsoft Teams so events can be shown or added directly from the site.

**Future Works – Adding More High-Tech Content Sources**

**1️. Blogs**

Add RSS feeds or API links from leading tech companies & research groups:

Google AI Blog – <https://ai.googleblog.com/>

Microsoft Research Blog – <https://www.microsoft.com/en-us/research/blog/>

Meta AI Blog – <https://ai.meta.com/blog/>

Amazon Science Blog – <https://www.amazon.science/blog>

NVIDIA Technical Blog – <https://developer.nvidia.com/blog/>

IBM Research Blog – <https://research.ibm.com/blog>

OpenAI Blog – <https://openai.com/blog>

DeepMind Blog – <https://deepmind.google/discover/blog/>

MIT Technology Review (Paid) – <https://www.technologyreview.com/>

**2️. Podcasts**

Include direct podcast show links or APIs where possible:

Google Cloud Platform Podcast – <https://cloud.google.com/blog/topics/podcasts>

Microsoft Research Podcast – <https://www.microsoft.com/en-us/research/podcast/>

AWS Podcast – <https://aws.amazon.com/podcasts/>

Meta Tech Podcast – (Some internal, but public interviews on YouTube/Spotify)

NVIDIA AI Podcast – <https://blogs.nvidia.com/ai-podcast/>

The Vergecast (Paid/Free Mix) – <https://www.theverge.com/the-vergecast-podcast>

Wired Tech in Two – <https://www.wired.com/category/podcasts/>

MIT Technology Review – In Machines We Trust (Paid) – <https://www.technologyreview.com/podcast/>

**3️. News**

High-authority tech news feeds for Industry News section:

Google News – Technology – <https://news.google.com/rss/topics/CAAqBwgKMJ6AmAswrtnkAw?oc=5>

Microsoft Blog – Official News – <https://blogs.microsoft.com/>

Meta Newsroom – <https://about.fb.com/news/>

Apple Newsroom – <https://www.apple.com/newsroom/>

NVIDIA Newsroom – <https://nvidianews.nvidia.com/news>

TechCrunch – <https://techcrunch.com/>(Paid for Pro)

The Verge – Tech – <https://www.theverge.com/tech>

Wired – Technology – <https://www.wired.com/category/tech/>

(Paid/Free Mix)

Bloomberg Technology (Paid) – <https://www.bloomberg.com/technology>

MIT Technology Review – News (Paid) – <https://www.technologyreview.com/>

**Detailed Troubleshooting Notes**

**1️. RSS Feeds (Blogs, Podcasts, Industry News)**

What Happened:

Some RSS feeds (like The Verge) did not return images, only text.

A few feeds were in XML but didn’t follow a consistent structure, causing parsing errors.

Some feeds timed out or blocked requests from the server because of missing headers.

What I Did to Fix:

For feeds missing images → switched to an alternate RSS source or used a paid API that included images.

For feeds with strict rules → added request headers to pretend like a normal browser request (User-Agent).

For feeds returning broken XML → ran them through an RSS-to-JSON service like rss2json to normalize the data.

**2️. Chatroom Integration**

What Happened:

Chatroom worked partially but messages didn’t always send/receive.

Connection errors popped up in the console (WebSocket connection failed or CORS error).

When chatbot and chatroom were both enabled, they sometimes interfered with each other.

What We Did to Fix:

Matched the frontend origin (site URL) with backend CORS settings so the WebSocket would accept connections.

Restarted both backend and Nginx after making changes — some errors were from outdated config.

Gave chatbot and chatroom separate namespaces so their connections wouldn’t mix.

**3️. SSO (Single Sign-On)**

What Happened:

SSO login didn’t work at all — no redirect, or redirect loop happened.

The Identity Provider (IdP) and Service Provider (SP) URLs didn’t match exactly.

The certificate in the server didn’t match the one registered with the IdP.

What We Did to Fix:

Checked the ACS URL (where the IdP sends the user after login) and made sure it matched exactly in both the IdP and our server config.

Updated the certificate file (.cer) to the correct one from the IdP.

Verified time sync on the server — a large time difference can break SSO.